KUROSE • ROSS

Computer Networking

A TOP-DOWN APPROACH

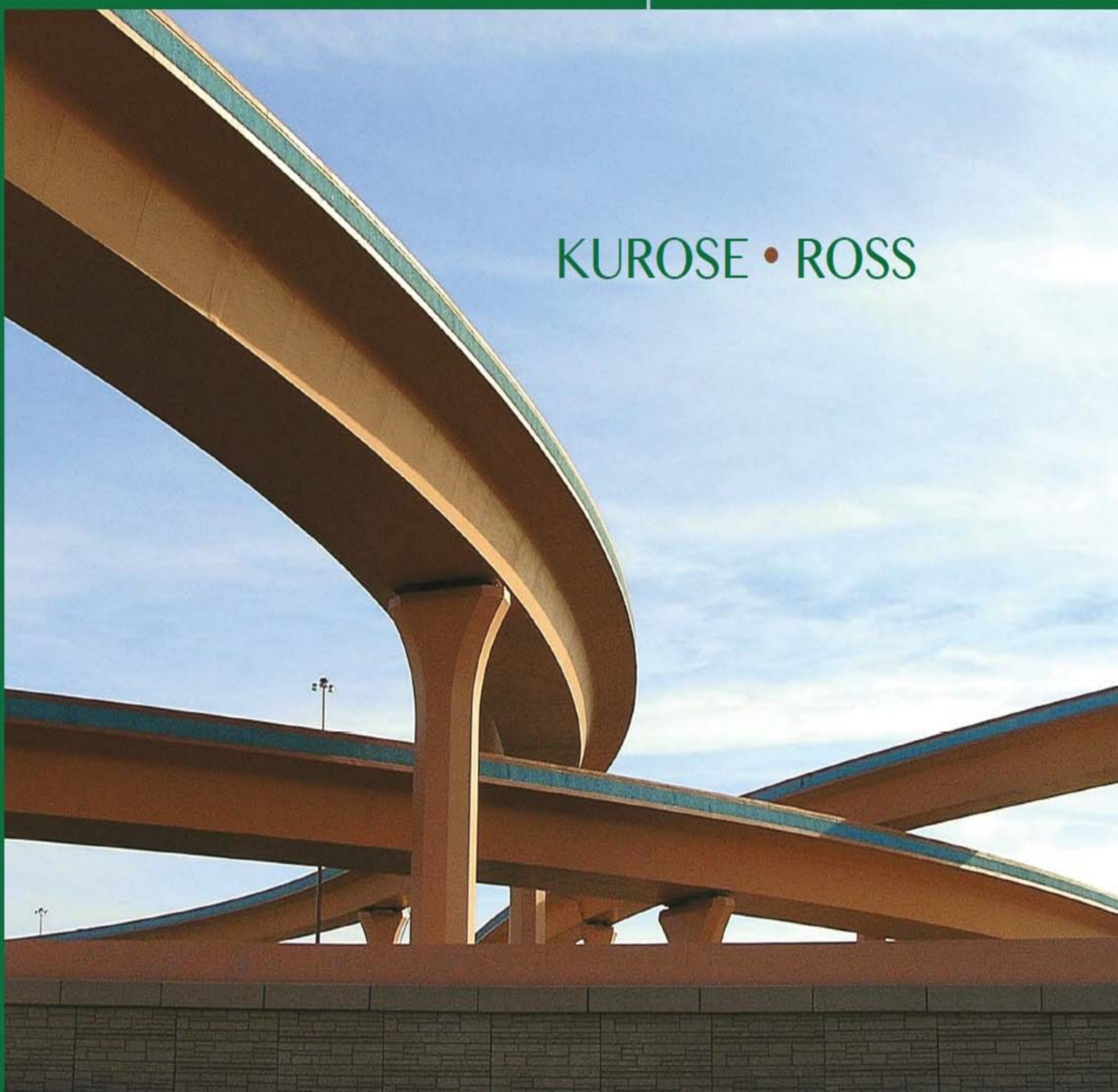
SEVENTH EDITION

PEARSON

Computer Networking

A TOP-DOWN APPROACH

SEVENTH EDITION



KUROSE • ROSS

कंप्यूटर नेटवर्किंग

ऊपर से नीचे तक का दृष्टिकोण

सातवां संस्करण

जेम्स एफ. कुरोसे

यूनिवर्सिटी ऑफ मैसाचुसेट्स, एमहर्स्ट कीथ डब्ल्यू. रॉस

NYU और NYU शंघाई

PEARSON

बोस्टन कोलंबस इंडियानापोलिस न्यूयॉर्क सैन फ्रांसिस्को होबोकेन एम्स्टर्डम केप
शहर दुबई लंदन मैड्रिड मिलान म्यूनिख पेरिस मॉन्ट्रियल टोरंटो दिल्ली मेक्सिको सिटी साओ
पाउलो सिडनी हांगकांग सियोल सिंगापुर ताइपे टोक्यो

उपाध्यक्ष, संपादकीय निदेशक, ईसीएस: मार्सिया हॉर्टन

अधिग्रहण संपादक: मैट गोल्डस्टीन

संपादकीय सहायक: क्रिस्टी अलौरा

विपणन उपाध्यक्ष: क्रिस्टी लेस्को

फील्ड मार्केटिंग निदेशक: टिम गैलिगन

उत्पाद विपणन प्रबंधक: ब्रैम वान केम्पेन

फील्ड मार्केटिंग मैनेजर: डेमेट्रियस हॉल

मार्केटिंग सहायक: जॉन ब्रायंट

उत्पाद प्रबंधन निदेशक: एरिन ग्रेग

टीम लीड, कार्यक्रम और परियोजना प्रबंधन: स्कॉट डिसानो

कार्यक्रम प्रबंधक: जोआन मैनिंग और कैरोल स्नाइडर

प्रोजेक्ट मैनेजर: कैटरीना ओस्लर, ओस्टलर एडिटोरियल, इंक.

वरिष्ठ विशेषज्ञ, कार्यक्रम योजना और समर्थन: मौरा ज़ाल्डिवार-गार्सिया

कवर डिजाइनर: जॉयस वेल्स

प्रबंधक, अधिकार और अनुमतियाँ: बेन फेरिनी

प्रोजेक्ट मैनेजर, अधिकार और अनुमतियाँ: जेनी हॉफमैन, अपटारा कॉर्पोरेशन

इन्वेंटरी मैनेजर: एन लैम

कवर छवि: मार्क गुटिरेज़/गेटी इमेजेज़

मीडिया प्रोजेक्ट मैनेजर: स्टीव राइट

रचना: सेनवेओ पब्लिशिंग सर्विसेज

प्रिंटर/बाइंडर: एडवर्ड्स ब्रदर्स मैलोय

कवर और इन्सर्ट प्रिंटर: फीनिक्स कलर/ हेगरस्टाउन

अन्य स्रोतों से उधार लिए गए तथा अनुमति से इस पाठ्यपुस्तक में पुनरुत्पादित श्रेय और आभार पाठ के भीतर उपयुक्त पृष्ठ पर दिए गए हैं।

लाइब्रेरी ऑफ कांग्रेस कैटलॉगिंग-इन-पब्लिकेशन डेटा

नाम: कुरोज़, जेम्स एफ. | रॉस, कीथ डब्ल्यू., 1956-

शीर्षक: कंप्यूटर नेटवर्किंग: एक टॉप-डाउन दृष्टिकोण / जेम्स एफ. कुरोस, यूनिवर्सिटी ऑफ मैसाचुसेट्स, एमहर्स्ट, कीथ डब्ल्यू. रॉस, एनवाईयू और एनवाईयू शंघाई।

विवरण: सातवाँ संस्करण। | होबोकेन, न्यू जर्सी: पियर्सन, [2017] | ग्रंथसूची संदर्भ और सूचकांक शामिल हैं।

पहचानकर्ता: LCCN 2016004976 | ISBN 9780133594140 | ISBN 0133594149

विषय: एलसीएसएच: इंटरनेट | कंप्यूटर नेटवर्क।

वर्गीकरण: LCC TK5105.875.I57 K88 2017 | DDC 004.6-dc23

LC रिकॉर्ड http://lccn.loc.gov/2016004976 **पर उपलब्ध है**

PEARSON

आईएसबीएन-10: 0-13-359414-9

आईएसबीएन-13: 978-0-13-359414-0

लेखकों के बारे में

जिम कुरोसे

जिम कुरोस मैसाचुसेट्स विश्वविद्यालय, एमहर्स्ट में कंप्यूटर विज्ञान के प्रतिष्ठित विश्वविद्यालय प्रोफेसर हैं। वे वर्तमान में मैसाचुसेट्स विश्वविद्यालय से छुट्टी पर हैं, और यूएस नेशनल साइंस फाउंडेशन में सहायक निदेशक के रूप में कार्य कर रहे हैं, जहाँ वे कंप्यूटर और सूचना विज्ञान और इंजीनियरिंग निदेशालय का नेतृत्व करते हैं।

डॉ. कुरोज़ को उनकी शैक्षणिक गतिविधियों के लिए कई सम्मान प्राप्त हुए हैं, जिनमें नेशनल टेक्नोलॉजिकल यूनिवर्सिटी (आठ बार), यूनिवर्सिटी ऑफ़ मैसाचुसेट्स और नॉर्थईस्ट एसोसिएशन ऑफ़ ग्रेजुएट स्कूल्स से उत्कृष्ट शिक्षक पुरस्कार शामिल हैं। उन्हें IEEE टेलर बूथ एजुकेशन मेडल मिला और मैसाचुसेट्स के कॉमनवेल्थ इंफॉर्मेशन टेक्नोलॉजी इनिशिएटिव के उनके नेतृत्व के लिए उन्हें सम्मानित किया गया। उन्होंने कई सम्मेलनों में सर्वश्रेष्ठ पेपर पुरस्कार जीते हैं और IEEE इन्फोकॉम अचीवमेंट अवार्ड और ACM सिगकॉम टेस्ट ऑफ़ टाइम अवार्ड प्राप्त किया है।



डॉ. कुरोज़ IEEE ट्रांजेक्शन ऑन कम्युनिकेशंस और IEEE/ACM के पूर्व प्रधान संपादक हैं

नेटवर्किंग पर लेनदेन। **उन्होंने** IEEE Infocom, ACM SIGCOMM, ACM इंटरनेट मापन सम्मेलन और ACM SIGMETRICS के लिए तकनीकी कार्यक्रम सह-अध्यक्ष के रूप में कार्य किया है ।
वह IEEE और ACM के फेलो हैं। उनके शोध हितों में नेटवर्क प्रोटोकॉल और वास्तुकला, नेटवर्क मापन, मल्टीमीडिया संचार और मॉडलिंग और प्रदर्शन मूल्यांकन शामिल हैं। उन्होंने कोलंबिया विश्वविद्यालय से कंप्यूटर विज्ञान में पीएचडी की है।

कीथ रॉस

कीथ रॉस NYU शंघाई में इंजीनियरिंग और कंप्यूटर विज्ञान के डीन हैं और NYU में कंप्यूटर विज्ञान और इंजीनियरिंग विभाग में लियोनार्ड जे. शस्टेक चेयर प्रोफेसर हैं। इससे पहले वे यूनिवर्सिटी ऑफ़ पेनसिल्वेनिया (13 वर्ष), यूरेकॉम इंस्टीट्यूट (5 वर्ष) और पॉलिटेक्निक यूनिवर्सिटी (10 वर्ष) में थे। उन्होंने टफ्ट्स यूनिवर्सिटी से BSEE, कोलंबिया यूनिवर्सिटी से MSEE और यूनिवर्सिटी ऑफ़ मिशिगन से कंप्यूटर और कंट्रोल इंजीनियरिंग में पीएचडी की डिग्री प्राप्त की है। कीथ रॉस विम्बा के सह-संस्थापक और मूल सीईओ भी हैं, जो ई-लर्निंग के लिए ऑनलाइन मल्टीमीडिया एप्लिकेशन विकसित करता है और जिसे 2010 में ब्लैकबोर्ड द्वारा अधिग्रहित किया गया था।



प्रोफेसर रॉस की शोध रुचि गोपनीयता, सामाजिक नेटवर्क, पीयर-टू-पीयर नेटवर्किंग, इंटरनेट मापन, सामग्री वितरण नेटवर्क और स्टोकेस्टिक मॉडलिंग में है। वह एक ACM फेलो, एक IEEE फेलो, इन्फोकॉम 2009 बेस्ट पेपर अवार्ड के प्राप्तकर्ता और मल्टीमीडिया संचार के लिए 2011 और 2008 के सर्वश्रेष्ठ पेपर अवार्ड (IEEE संचार सोसायटी द्वारा सम्मानित) के प्राप्तकर्ता हैं। उन्होंने IEEE/ACM ट्रांजेक्शन ऑन नेटवर्किंग, ACM SIGCOMM, ACM CoNext और ACM इंटरनेट मापन सम्मेलन सहित कई जर्नल संपादकीय बोर्डों और सम्मेलन कार्यक्रम समितियों में काम किया है। उन्होंने P2P फ़ाइल शेयरिंग पर संघीय व्यापार आयोग के सलाहकार के रूप में भी काम किया है।

जूली और हमारे तीन प्यारे लोगों - क्रिस, चार्ली और नीना को

जेकेएफ़

दुनिया भर के मेरे प्रोफेसरों, सहकर्मियों और छात्रों को बहुत-बहुत धन्यवाद।

केडब्ल्यूआर

प्रस्तावना

कंप्यूटर नेटवर्किंग: ए टॉप-डाउन अप्रोच के सातवें संस्करण में आपका स्वागत है । 16 साल पहले पहले संस्करण के प्रकाशन के बाद से, हमारी पुस्तक को सैकड़ों कॉलेजों और विश्वविद्यालयों में उपयोग के लिए अपनाया गया है, 14 भाषाओं में अनुवादित किया गया है, और दुनिया भर में एक लाख से अधिक छात्रों और चिकित्सकों द्वारा उपयोग किया गया है।
हमने इनमें से कई पाठकों से सुना है और सकारात्मक प्रतिक्रिया से अभिभूत हैं।

सातवें संस्करण में नया क्या है?

हमें लगता है कि इस सफलता का एक महत्वपूर्ण कारण यह है कि हमारी पुस्तक कंप्यूटर नेटवर्किंग निर्देश के लिए एक नया और समयोचित दृष्टिकोण प्रदान करती है। हमने इस सातवें संस्करण में बदलाव किए हैं, लेकिन हमने उन बातों को भी अपरिवर्तित रखा है जिन्हें हम मानते हैं (और हमारे पुस्तक का उपयोग करने वाले प्रशिक्षकों और छात्रों ने पुष्टि की है) कि वे इस पुस्तक के सबसे महत्वपूर्ण पहलू हैं: इसका शीर्ष-से-नीचे का दृष्टिकोण, इंटरनेट पर इसका ध्यान और कंप्यूटर नेटवर्किंग का आधुनिक उपचार, सिद्धांतों और अभ्यास दोनों पर इसका ध्यान, और कंप्यूटर नेटवर्किंग के बारे में सीखने के प्रति इसकी सुलभ शैली और दृष्टिकोण। फिर भी, सातवें संस्करण को काफी हद तक संशोधित और अद्यतन किया गया है।

हमारी पुस्तक के पुराने पाठक देखेंगे कि इस पाठ के प्रकाशित होने के बाद पहली बार हमने अध्यायों के संगठन को ही बदल दिया है। नेटवर्क लेयर, जिसे पहले एक ही अध्याय में कवर किया गया था, अब अध्याय 4 (जो नेटवर्क लेयर के तथाकथित "डेटा प्लेन" घटक पर केंद्रित है) और अध्याय 5 (जो नेटवर्क लेयर के "कंट्रोल प्लेन" पर केंद्रित है) में शामिल है। नेटवर्क लेयर का यह विस्तारित कवरेज सॉफ़्टवेयर-परिभाषित नेटवर्किंग (SDN) के महत्व में तेज़ी से वृद्धि को दर्शाता है, जो यकीनन दशकों में नेटवर्किंग में सबसे महत्वपूर्ण और रोमांचक प्रगति है। हालाँकि यह एक अपेक्षाकृत हालिया नवाचार है, लेकिन SDN को व्यवहार में तेज़ी से अपनाया गया है - इतना कि आधुनिक कंप्यूटर नेटवर्किंग के ऐसे परिचय की कल्पना करना पहले से ही कठिन है जिसमें SDN को शामिल न किया गया हो। नेटवर्क प्रबंधन का विषय, जिसे पहले अध्याय 9 में कवर किया गया था, अब नए अध्याय 5 में शामिल कर दिया गया है। हमेशा की तरह, हमने छठे संस्करण के बाद से नेटवर्किंग के गतिशील क्षेत्र में हाल के परिवर्तनों को दर्शाने के लिए पाठ के कई अन्य खंडों को भी अपडेट किया है। हमेशा की तरह, मुद्रित पाठ से हटाई गई सामग्री हमेशा इस पुस्तक की सहयोगी वेबसाइट पर पाई जा सकती है।

सबसे महत्वपूर्ण अपडेट निम्नलिखित हैं:

- अध्याय 1 को इंटरनेट की बढ़ती पहुंच और उपयोग को प्रतिबिंबित करने के लिए अद्यतन किया गया है।
- अध्याय 2, जो एप्लीकेशन लेयर को कवर करता है, को महत्वपूर्ण रूप से अपडेट किया गया है। हमने एप्लीकेशन-लेवल वीडियो स्ट्रीमिंग और कंटेंट डिस्ट्रीब्यूशन नेटवर्क पर एक नए सेक्शन के लिए जगह बनाने के लिए FTP प्रोटोकॉल और वितरित हैश टेबल पर सामग्री को हटा दिया है, साथ ही नेटफ्लिक्स और यूट्यूब केस स्टडीज़ भी शामिल की हैं। सॉकेट प्रोग्रामिंग सेक्शन को पायथन 2 से पायथन 3 में अपडेट किया गया है।

- अध्याय 3, जो ट्रांसपोर्ट लेयर को कवर करता है, को मामूली रूप से अपडेट किया गया है। एसिंक्रोनस ट्रांसपोर्ट मोड (एटीएम) नेटवर्क पर सामग्री को इंटरनेट की स्पष्ट भीड़ अधिसूचना (ईसीएन) पर अधिक आधुनिक सामग्री द्वारा प्रतिस्थापित किया गया है, जो समान सिद्धांतों को सिखाता है।

- अध्याय 4 नेटवर्क परत के "डेटा प्लेन" घटक को कवर करता है - प्रति-राउटर फ़ॉरवर्डिंग फ़ंक्शन जो यह निर्धारित करता है कि राउटर के इनपुट लिंक में से किसी एक पर आने वाले पैकेट को उस राउटर के आउटपुट लिंक में से किसी एक पर कैसे फ़ॉरवर्ड किया जाए। हमने सभी पिछले संस्करणों में पाए जाने वाले पारंपरिक इंटरनेट फ़ॉरवर्डिंग पर सामग्री को अपडेट किया है, और पैकेट शेड्यूलिंग पर सामग्री जोड़ी है। हमने SDN में प्रचलित सामान्यीकृत फ़ॉरवर्डिंग पर एक नया खंड भी जोड़ा है। पूरे अध्याय में कई अपडेट भी हैं। नई सामग्री के लिए जगह बनाने के लिए मल्टीकास्ट और ब्रॉडकास्ट संचार पर सामग्री को हटा दिया गया है।

- अध्याय 5 में , हम नेटवर्क परत के नियंत्रण तल कार्यों को कवर करते हैं - नेटवर्क-व्यापी तर्क जो नियंत्रित करता है कि स्रोत होस्ट से गंतव्य होस्ट तक राउटर के अंत-से-अंत पथ के साथ डेटाग्राम को कैसे रूट किया जाता है। पिछले संस्करणों की तरह, हम रूटिंग एल्गोरिदम, साथ ही आज के इंटरनेट में उपयोग किए जाने वाले रूटिंग प्रोटोकॉल (BGP के अपडेटेड ट्रीटमेंट के साथ) को कवर करते हैं। हमने SDN नियंत्रण तल पर एक महत्वपूर्ण नया खंड जोड़ा है, जहाँ रूटिंग और अन्य फ़ंक्शन तथाकथित SDN नियंत्रकों में कार्यान्वित किए जाते हैं।

- अध्याय 6, जो अब लिंक परत को कवर करता है, में ईथरनेट और डेटा सेंटर नेटवर्किंग का अद्यतन विवरण दिया गया है।

- अध्याय 7, जो वायरलेस और मोबाइल नेटवर्किंग को कवर करता है, में 802.11 (तथाकथित "वाईफाई) नेटवर्क और 4 जी और एलटीई सहित सेलुलर नेटवर्क पर अद्यतन सामग्री शामिल है।
- अध्याय 8, जो नेटवर्क सुरक्षा को कवर करता है और जिसे छठे संस्करण में बड़े पैमाने पर अद्यतन किया गया था, में केवल

इस सातवें संस्करण में कुछ मामूली अपडेट किये गये हैं।

- **मल्टीमीडिया नेटवर्किंग पर** अध्याय 9, अब छठे संस्करण की तुलना में थोड़ा "पतला" है, क्योंकि वीडियो स्ट्रीमिंग और सामग्री वितरण नेटवर्क पर सामग्री को अध्याय 2 में स्थानांतरित कर दिया गया है, और पैकेट शेड्यूलिंग पर सामग्री को अध्याय 4 में शामिल किया गया है।

- अध्याय के अंत में आने वाली समस्याओं से संबंधित महत्वपूर्ण नई सामग्री जोड़ी गई है। पिछले सभी संस्करणों की तरह, होमवर्क की समस्याओं को संशोधित किया गया है, जोड़ा गया है और हटाया गया है।

हमेशा की तरह, हमारी पुस्तक के इस नए संस्करण को बनाने का हमारा उद्देश्य कंप्यूटर नेटवर्किंग पर केंद्रित और आधुनिक उपचार प्रदान करना जारी रखना है, जिसमें सिद्धांतों और व्यवहार दोनों पर जोर दिया गया है।

श्रोता

यह पाठ्यपुस्तक कंप्यूटर नेटवर्किंग पर पहले कोर्स के लिए है। इसका उपयोग कंप्यूटर विज्ञान और इलेक्ट्रिकल इंजीनियरिंग दोनों विभागों में किया जा सकता है। प्रोग्रामिंग भाषाओं के संदर्भ में, पुस्तक केवल यह मानती है कि छात्र को C, C++, Java, या Python (और तब भी केवल कुछ स्थानों पर) का अनुभव है। हालाँकि यह पुस्तक कई अन्य परिचयात्मक कंप्यूटर नेटवर्किंग पाठ्य पुस्तकों की तुलना में अधिक सटीक और विश्लेषणात्मक है, लेकिन इसमें शायद ही कभी किसी गणितीय अवधारणा का उपयोग किया गया हो जो हाई स्कूल में नहीं पढ़ाई जाती है। हमने किसी भी उन्नत कलन, प्रायिकता या स्टोकेस्टिक प्रक्रिया अवधारणाओं का उपयोग करने से बचने का जानबूझकर प्रयास किया है (हालाँकि हमने इस उन्नत पृष्ठभूमि वाले छात्रों के लिए कुछ गृहकार्य समस्याएँ शामिल की हैं)। इसलिए यह पुस्तक स्नातक पाठ्यक्रमों और प्रथम वर्ष के स्नातकोत्तर पाठ्यक्रमों के लिए उपयुक्त है। यह दूरसंचार उद्योग में काम करने वालों के लिए भी उपयोगी होनी चाहिए।

इस पाठ्यपुस्तक की क्या विशेषता है?

कंप्यूटर नेटवर्किंग का विषय बहुत जटिल है, जिसमें कई अवधारणाएँ, प्रोटोकॉल और प्रौद्योगिकियाँ शामिल हैं जो एक जटिल तरीके से एक साथ बुनी गई हैं। इस दायरे और जटिलता से निपटने के लिए, कई कंप्यूटर नेटवर्किंग पाठ अक्सर नेटवर्क आर्किटेक्चर की "परतों" के आसपास व्यवस्थित होते हैं। एक स्तरित संगठन के साथ, छात्र कंप्यूटर नेटवर्किंग की जटिलता को देख सकते हैं - वे आर्किटेक्चर के एक हिस्से में अलग-अलग अवधारणाओं और प्रोटोकॉल के बारे में सीखते हैं जबकि सभी भागों को एक साथ कैसे फिट किया जाता है, इसकी बड़ी तस्वीर देखते हैं।

शैक्षणिक दृष्टिकोण से, हमारा व्यक्तिगत अनुभव यह रहा है कि इस तरह का स्तरित दृष्टिकोण वास्तव में अच्छा काम करता है। फिर भी, हमने पाया है कि शिक्षण का पारंपरिक तरीका - नीचे से ऊपर; यानी भौतिक परत से अनुप्रयोग परत की ओर - कंप्यूटर नेटवर्किंग पर आधुनिक पाठ्यक्रम के लिए सबसे अच्छा तरीका नहीं है।

ऊपर से नीचे तक का दृष्टिकोण

हमारी किताब ने 16 साल पहले नेटवर्किंग को टॉप-डाउन तरीके से पेश करके नई राह खोली थी - यानी, एप्लीकेशन लेयर से शुरू करके और फिजिकल लेयर की ओर नीचे की ओर काम करके। शिक्षकों और छात्रों से हमें जो प्रतिक्रिया मिली, उसने पुष्टि की है कि इस टॉप-डाउन दृष्टिकोण के कई फायदे हैं और यह वास्तव में शैक्षणिक रूप से अच्छा काम करता है। सबसे पहले, यह एप्लीकेशन लेयर (नेटवर्किंग में एक "उच्च विकास क्षेत्र") पर जोर देता है। वास्तव में, कंप्यूटर नेटवर्किंग में हाल ही में हुई कई क्रांतियाँ - जिसमें वेब, पीयर-टू-पीयर फ़ाइल शेयरिंग और मीडिया स्ट्रीमिंग शामिल हैं - एप्लीकेशन लेयर पर हुई हैं। एप्लीकेशन-लेयर मुद्दों पर शुरुआती जोर अधिकांश अन्य ग्रंथों में अपनाए गए दृष्टिकोणों से अलग है, जिनमें नेटवर्क अनुप्रयोगों, उनकी आवश्यकताओं, एप्लीकेशन-लेयर प्रतिमानों (जैसे, क्लाइंट-सर्वर और पीयर-टू-पीयर) और एप्लीकेशन प्रोग्रामिंग इंटरफेस पर बहुत कम सामग्री है। दूसरा, प्रशिक्षकों के रूप में हमारा अनुभव (और इस पाठ का उपयोग करने वाले कई प्रशिक्षकों का) यह रहा है कि पाठ्यक्रम की शुरुआत में नेटवर्किंग अनुप्रयोगों को पढ़ाना एक शक्तिशाली प्रेरक उपकरण है। छात्र यह जानकर रोमांचित होते हैं कि नेटवर्किंग कैसे काम करती है

अनुप्रयोग कार्य करते हैं - ईमेल और वेब जैसे अनुप्रयोग, जिनका अधिकांश छात्र दैनिक आधार पर उपयोग करते हैं। एक बार जब कोई छात्र अनुप्रयोगों को समझ लेता है, तो वह इन अनुप्रयोगों का समर्थन करने के लिए आवश्यक नेटवर्क सेवाओं को समझ सकता है। फिर छात्र, बदले में, विभिन्न तरीकों की जांच कर सकता है जिससे ऐसी सेवाएं प्रदान की जा सकती हैं और निचली परतों में कार्यान्वित की जा सकती हैं। इस प्रकार अनुप्रयोगों को पहले से कवर करना पाठ के शेष भाग के लिए प्रेरणा प्रदान करता है।

तीसरा, टॉप-डाउन दृष्टिकोण प्रशिक्षकों को शुरुआती चरण में नेटवर्क एप्लिकेशन डेवलपमेंट शुरू करने में सक्षम बनाता है। छात्र न केवल यह देखते हैं कि लोकप्रिय एप्लिकेशन और प्रोटोकॉल कैसे काम करते हैं, बल्कि यह भी सीखते हैं कि अपने स्वयं के नेटवर्क एप्लिकेशन और एप्लिकेशन-स्तरीय प्रोटोकॉल बनाना कितना आसान है। टॉप-डाउन दृष्टिकोण के साथ, छात्रों को सॉकेट प्रोग्रामिंग, सेवा मॉडल और प्रोटोकॉल की अवधारणाओं के बारे में जल्दी जानकारी मिलती है - महत्वपूर्ण अवधारणाएँ जो बाद की सभी परतों में फिर से उभरती हैं। पायथन में सॉकेट प्रोग्रामिंग के उदाहरण प्रदान करके, हम छात्रों को जटिल कोड से भ्रमित किए बिना केंद्रीय विचारों को उजागर करते हैं। इलेक्ट्रिकल इंजीनियरिंग और कंप्यूटर विज्ञान में स्नातक छात्रों को पायथन कोड का पालन करने में कठिनाई नहीं होनी चाहिए।

इंटरनेट पर ध्यान

हालाँकि हमने चौथे संस्करण के साथ इस पुस्तक के शीर्षक से “फीचरिंग द इंटरनेट” वाक्यांश हटा दिया है, लेकिन इसका मतलब यह नहीं है कि हमने इंटरनेट पर अपना ध्यान केंद्रित करना छोड़ दिया है। वास्तव में, इससे ज़्यादा कुछ भी नहीं हो सकता!
इसके बजाय, चूंकि इंटरनेट इतना व्यापक हो गया है, इसलिए हमें लगा कि किसी भी नेटवर्किंग पाठ्यपुस्तक में इंटरनेट पर महत्वपूर्ण ध्यान दिया जाना चाहिए, और इस प्रकार यह वाक्यांश कुछ हद तक अनावश्यक था। हम मौलिक कंप्यूटर नेटवर्किंग अवधारणाओं का अध्ययन करने के लिए प्राथमिक वाहनों के रूप में इंटरनेट की वास्तुकला और प्रोटोकॉल का उपयोग करना जारी रखते हैं। बेशक, हम अन्य नेटवर्क आर्किटेक्चर से अवधारणाओं और प्रोटोकॉल को भी शामिल करते हैं। लेकिन स्पॉटलाइट स्पष्ट रूप से इंटरनेट पर है, एक तथ्य जो इंटरनेट की पांच-परत वास्तुकला के आसपास पुस्तक को व्यवस्थित करने में परिलक्षित होता है: एप्लिकेशन, ट्रांसपोर्ट, नेटवर्क, लिंक और भौतिक परतें।

इंटरनेट पर प्रकाश डालने का एक और लाभ यह है कि अधिकांश कंप्यूटर विज्ञान और इलेक्ट्रिकल इंजीनियरिंग के छात्र इंटरनेट और इसके प्रोटोकॉल के बारे में जानने के लिए उत्सुक हैं। वे जानते हैं कि इंटरनेट एक क्रांतिकारी और विघटनकारी तकनीक रही है और वे देख सकते हैं कि यह हमारी दुनिया को गहराई से बदल रही है। इंटरनेट की अत्यधिक प्रासंगिकता को देखते हुए, छात्र स्वाभाविक रूप से इस बारे में उत्सुक हैं कि "हुड के नीचे क्या है।" इस प्रकार, एक प्रशिक्षक के लिए इंटरनेट को मार्गदर्शक फोकस के रूप में उपयोग करते समय छात्रों को बुनियादी सिद्धांतों के बारे में उत्साहित करना आसान है।

नेटवर्किंग सिद्धांतों का शिक्षण

पुस्तक की दो अनूठी विशेषताएँ - इसका शीर्ष-से-नीचे का दृष्टिकोण और इंटरनेट पर इसका ध्यान - हमारी पुस्तक के शीर्षकों में दिखाई दिए हैं। यदि हम उपशीर्षक में तीसरा **वाक्यांश जोड़ सकते थे, तो इसमें** सिद्धांत शब्द शामिल होता। नेटवर्किंग का क्षेत्र अब इतना परिपक्व हो गया है कि कई मौलिक रूप से महत्वपूर्ण मुद्दों की पहचान की जा सकती है। उदाहरण के लिए, ट्रांसपोर्ट लेयर में, मूलभूत मुद्दों में एक अविश्वसनीय नेटवर्क लेयर पर विश्वसनीय संचार, कनेक्शन स्थापना / टियरडाउन और हैंडशेकिंग, भीड़भाड़ और प्रवाह नियंत्रण, और मल्टीप्लेक्सिंग शामिल हैं। तीन मौलिक रूप से महत्वपूर्ण नेटवर्क-लेयर मुद्दे दो राउटर के बीच "अच्छे" पथों का निर्धारण, बड़ी संख्या में विषम नेटवर्क को आपस में जोड़ना और एक आधुनिक नेटवर्क की जटिलता का प्रबंधन करना है। लिंक लेयर में, एक मौलिक समस्या एकाधिक एक्सेस चैनल को साझा करना है।

नेटवर्क सुरक्षा में, गोपनीयता, प्रमाणीकरण और संदेश अखंडता प्रदान करने की तकनीकें सभी क्रिप्टोग्राफ़िक मूल सिद्धांतों पर आधारित हैं। यह पाठ मौलिक नेटवर्किंग मुद्दों की पहचान करता है और इन मुद्दों को संबोधित करने के तरीकों का अध्ययन करता है। इन सिद्धांतों को सीखने वाले छात्र को लंबे समय तक "शेल्फ लाइफ" वाला ज्ञान प्राप्त होगा - आज के नेटवर्क मानकों और प्रोटोकॉल के अप्रचलित हो जाने के बहुत बाद तक, वे सिद्धांत महत्वपूर्ण और प्रासंगिक बने रहेंगे। हमारा मानना है कि छात्र को दरवाज़े पर लाने के लिए इंटरनेट का उपयोग करने और फिर मौलिक मुद्दों और समाधान दृष्टिकोणों पर जोर देने का संयोजन छात्र को अनुमति देगा

किसी भी नेटवर्किंग तकनीक को शीघ्रता से समझें।

वेबसाइट

इस पाठ्यपुस्तक की प्रत्येक नई प्रति में सभी पुस्तक पाठकों के लिए http://www.pearsonhighered.com/cs-resources/ पर बारह महीने तक सहयोगी वेबसाइट तक पहुंच शामिल है। जो भी शामिल है:

- इंटरैक्टिव शिक्षण सामग्री। पुस्तक की सहयोगी वेबसाइट में वीडियो नोट्स शामिल हैं - पुस्तक में लेखकों द्वारा किए गए महत्वपूर्ण विषयों के वीडियो प्रस्तुतीकरण, साथ ही अध्याय के अंत में समस्याओं के समान समाधानों के वॉकथ्रू। हमने अध्याय 1 से 5 **के लिए वीडियो नोट्स और ऑनलाइन समस्याओं के साथ वेबसाइट को तैयार किया है** और समय के साथ इस सामग्री को सक्रिय रूप से जोड़ना और अपडेट करना जारी रखेंगे। पिछले संस्करणों की तरह, वेबसाइट में इंटरैक्टिव जावा एप्लेट शामिल हैं जो कई प्रमुख नेटवर्किंग अवधारणाओं को एनिमेट करते हैं। साइट में इंटरैक्टिव क्विज़ भी हैं जो छात्रों को विषय वस्तु की अपनी बुनियादी समझ की जाँच करने की अनुमति देते हैं। प्रोफेसर इन इंटरैक्टिव सुविधाओं को अपने पाठ्यक्रम में एकीकृत कर सकते हैं।

व्याख्यानों में भाग लें या उन्हें मिनी प्रयोगशाला के रूप में उपयोग करें।

- अतिरिक्त तकनीकी सामग्री। चूँकि हमने अपनी पुस्तक के प्रत्येक संस्करण में नई सामग्री जोड़ी है, इसलिए हमें पुस्तक की लंबाई को नियंत्रित रखने के लिए कुछ मौजूदा विषयों का कवरेज हटाना पड़ा है। उदाहरण के लिए, इस संस्करण में नई सामग्री के लिए जगह बनाने के लिए, हमने FTP, वितरित हैश टेबल और मल्टीकास्टिंग पर सामग्री हटा दी है, पाठ के पिछले संस्करणों में दिखाई देने वाली सामग्री अभी भी रुचिकर है, और इसलिए इसे पाया जा सकता है

  पुस्तक की वेब साइट पर.
- प्रोग्रामिंग असाइनमेंट। वेब साइट कई विस्तृत प्रोग्रामिंग असाइनमेंट भी प्रदान करती है, जिसमें एक मल्टीथ्रेडेड वेब सर्वर का निर्माण, एक GUI इंटरफ़ेस के साथ एक ई-मेल क्लाइंट का निर्माण, एक विश्वसनीय डेटा ट्रांसपोर्ट प्रोटोकॉल के प्रेषक और रिसीवर पक्षों को प्रोग्रामिंग करना, एक वितरित रूटिंग एल्गोरिदम प्रोग्रामिंग करना, और बहुत कुछ शामिल है।

- वायरशार्क प्रयोगशालाएँ। नेटवर्क प्रोटोकॉल को क्रियान्वित होते देखकर उनकी समझ काफ़ी गहरी हो सकती है। वेबसाइट पर वायरशार्क के कई असाइनमेंट दिए गए हैं, जो छात्रों को दो प्रोटोकॉल इकाइयों के बीच आदान-प्रदान किए जाने वाले संदेशों के अनुक्रम को वास्तव में देखने में सक्षम बनाते हैं। वेबसाइट में HTTP, DNS, TCP, UDP, IP, ICMP, Ethernet, ARP, WiFi, SSL, और वेब पेज को प्राप्त करने के अनुरोध को पूरा करने में शामिल सभी प्रोटोकॉल का पता लगाने पर अलग-अलग वायरशार्क प्रयोगशालाएँ शामिल हैं। हम समय के साथ नई प्रयोगशालाएँ जोड़ते रहेंगे।

साथी वेबसाइट के अलावा, लेखक एक सार्वजनिक वेबसाइट http://gaia.cs.umass.edu/kurose_ross/interactive भी रखते हैं, जिसमें इंटरैक्टिव अभ्यास शामिल हैं जो अध्याय के अंत में चुनी गई समस्याओं के समान समस्याएँ बनाते हैं (और उनके लिए समाधान प्रस्तुत करते हैं)। चूँकि छात्र असीमित संख्या में समान समस्या उदाहरण बना सकते हैं (और उनके लिए समाधान देख सकते हैं), वे तब तक काम कर सकते हैं जब तक कि सामग्री वास्तव में समझ में न आ जाए।

महारत हासिल.

शैक्षणिक विशेषताएँ

हम में से प्रत्येक 30 से अधिक वर्षों से कंप्यूटर नेटवर्किंग पढ़ा रहे हैं। साथ मिलकर, हम इस पाठ में 60 से अधिक वर्षों का शिक्षण अनुभव लाते हैं, जिसके दौरान हमने कई हज़ार छात्रों को पढ़ाया है। हम इस दौरान कंप्यूटर नेटवर्किंग में सक्रिय शोधकर्ता भी रहे हैं। (वास्तव में, जिम और कीथ पहली बार 1979 में कोलंबिया विश्वविद्यालय में मिशा श्वार्टज़ द्वारा पढ़ाए गए कंप्यूटर नेटवर्किंग पाठ्यक्रम में मास्टर के छात्रों के रूप में एक-दूसरे से मिले थे।) हमें लगता है कि यह सब हमें इस बारे में एक अच्छा दृष्टिकोण देता है कि नेटवर्किंग कहाँ रही है और भविष्य में यह कहाँ जा सकती है। फिर भी, हमने इस पुस्तक में सामग्री को अपने स्वयं के पसंदीदा शोध परियोजनाओं के प्रति पक्षपाती बनाने के प्रलोभनों का विरोध किया है। हमें लगता है कि यदि आप हमारे शोध में रुचि रखते हैं तो आप हमारी व्यक्तिगत वेब साइटों पर जा सकते हैं। इस प्रकार, यह पुस्तक आधुनिक कंप्यूटर नेटवर्किंग के बारे में है - यह समकालीन प्रोटोकॉल और प्रौद्योगिकियों के साथ-साथ इन प्रोटोकॉल और प्रौद्योगिकियों के पीछे अंतर्निहित सिद्धांतों के बारे में है। हम यह भी मानते हैं

नेटवर्किंग के बारे में सीखना (और सिखाना!) मज़ेदार हो सकता है। इस पुस्तक में हास्य की भावना, उपमाओं का उपयोग और वास्तविक दुनिया के उदाहरण उम्मीद है कि इस सामग्री को और अधिक मज़ेदार बना देंगे।

प्रशिक्षकों के लिए अनुपूरक

हम इस कोर्स को पढ़ाने में प्रशिक्षकों की सहायता के लिए एक पूर्ण पूरक पैकेज प्रदान करते हैं। इस सामग्री को पियर्सन के प्रशिक्षक संसाधन केंद्र (http://www.pearsonhighered.com/irc) से प्राप्त किया जा सकता है। इन प्रशिक्षक पूरकों तक पहुंचने के बारे में जानकारी के लिए प्रशिक्षक संसाधन केंद्र पर जाएं।

- पावरपॉइंट स्लाइड्स। हम® सभी नौ अध्यायों के लिए पावरपॉइंट स्लाइड्स प्रदान करते हैं। इस सातवें संस्करण के साथ स्लाइड्स को पूरी तरह से अपडेट किया गया है। स्लाइड्स प्रत्येक अध्याय को विस्तार से कवर करती हैं। वे स्लाइड्स को रोचक और नेत्रहीन रूप से आकर्षक बनाने के लिए ग्राफ़िक्स और एनिमेशन (केवल नीरस टेक्स्ट बुलेट पर निर्भर रहने के बजाय) का उपयोग करते हैं। हम मूल पावरपॉइंट स्लाइड्स प्रदान करते हैं ताकि आप उन्हें अपनी शिक्षण आवश्यकताओं के अनुसार अनुकूलित कर सकें। इनमें से कुछ स्लाइड्स अन्य प्रशिक्षकों द्वारा योगदान की गई हैं जिन्होंने हमारी पुस्तक से पढ़ाया है।

- होमवर्क समाधान। हम पाठ्य सामग्री, प्रोग्रामिंग असाइनमेंट और वायरशार्क प्रयोगशालाओं में होमवर्क समस्याओं के लिए समाधान पुस्तिका प्रदान करते हैं। जैसा कि पहले उल्लेख किया गया है, हमने पुस्तक के पहले छह अध्यायों में कई नई होमवर्क समस्याएं पेश की हैं।

अध्याय निर्भरताएँ

इस पाठ का पहला अध्याय कंप्यूटर नेटवर्किंग का एक स्व-निहित अवलोकन प्रस्तुत करता है। कई प्रमुख अवधारणाओं और शब्दावली का परिचय देते हुए, यह अध्याय पुस्तक के बाकी हिस्सों के लिए मंच तैयार करता है। अन्य सभी अध्याय सीधे इस पहले अध्याय पर निर्भर हैं। अध्याय 1 को पूरा करने के बाद, हम प्रशिक्षकों को **हमारे शीर्ष-डाउन दर्शन का पालन करते हुए,** अध्याय 2 से 6 को क्रम में कवर करने की सलाह देते हैं। इन पाँचों अध्यायों में से प्रत्येक पिछले अध्यायों की सामग्री का लाभ उठाता है। पहले छह अध्यायों को पूरा करने के बाद, प्रशिक्षक के पास काफी लचीलापन होता है। अंतिम तीन अध्यायों के बीच कोई अंतर-निर्भरता नहीं है, इसलिए उन्हें किसी भी क्रम में पढ़ाया जा सकता है। हालाँकि, अंतिम तीन अध्यायों में से प्रत्येक पहले छह अध्यायों की सामग्री पर निर्भर करता है। कई प्रशिक्षक पहले पहले छह अध्यायों को पढ़ाते हैं और फिर "मिठाई" के लिए अंतिम तीन अध्यायों में से एक को पढ़ाते हैं।

एक अंतिम टिप्पणी: हमें आपसे सुनना अच्छा लगेगा

हम छात्रों और प्रशिक्षकों को प्रोत्साहित करते हैं कि वे हमारी पुस्तक के बारे में अपनी कोई भी टिप्पणी हमें ई-मेल से भेजें। हमारे पहले पाँच संस्करणों के बारे में दुनिया भर के इतने सारे प्रशिक्षकों और छात्रों से सुनना हमारे लिए अद्भुत रहा है। हमने इनमें से कई सुझावों को पुस्तक के बाद के संस्करणों में शामिल किया है। हम प्रशिक्षकों को हमें नई गृहकार्य समस्याएँ (और समाधान) भेजने के लिए भी प्रोत्साहित करते हैं जो वर्तमान गृहकार्य समस्याओं के पूरक होंगे। हम इन्हें वेबसाइट के केवल प्रशिक्षक वाले हिस्से पर पोस्ट करेंगे। हम प्रशिक्षकों और छात्रों को नए जावा एप्लेट बनाने के लिए भी प्रोत्साहित करते हैं जो इस पुस्तक में अवधारणाओं और प्रोटोकॉल को दर्शाते हैं। यदि आपके पास कोई एप्लेट है जो आपको लगता है कि इस पाठ के लिए उपयुक्त होगा, तो कृपया इसे हमें भेजें। यदि एप्लेट (नोटेशन और शब्दावली सहित) उपयुक्त है, तो हम इसे एप्लेट के लेखकों के उचित संदर्भ के साथ पाठ की वेबसाइट पर शामिल करने में प्रसन्न होंगे।

तो, जैसा कि कहावत है, "कार्ड और पत्र भेजते रहें!" गंभीरता से, कृपया **हमें दिलचस्प URL भेजना जारी** रखें , टाइपो को इंगित करें, हमारे किसी भी दावे से असहमत हों, और हमें बताएं कि क्या काम करता है और क्या काम नहीं करता है। हमें बताएं कि आपको क्या लगता है कि अगले संस्करण में क्या शामिल किया जाना चाहिए या नहीं। अपना ई-मेल kurose@cs.umass.edu **पर भेजें** और keithwross@nyu.edu.

स्वीकृतियाँ

1996 में जब से हमने इस पुस्तक को लिखना शुरू किया है, तब से कई लोगों ने हमें अमूल्य सहायता दी है और नेटवर्किंग पाठ्यक्रम को सर्वोत्तम तरीके से व्यवस्थित करने और पढ़ाने के बारे में हमारे विचारों को आकार देने में प्रभावशाली रहे हैं। हम उन सभी को बहुत-बहुत धन्यवाद कहना चाहते हैं जिन्होंने इस पुस्तक के शुरुआती ड्राफ्ट से लेकर इस सातवें संस्करण तक हमारी मदद की है। हम दुनिया भर के सैकड़ों पाठकों - छात्रों, शिक्षकों, चिकित्सकों - के भी बहुत आभारी हैं जिन्होंने हमें पुस्तक के पिछले संस्करणों पर विचार और टिप्पणियाँ भेजी हैं और पुस्तक के भविष्य के संस्करणों के लिए सुझाव दिए हैं। विशेष धन्यवाद:

अल अहो (कोलंबिया विश्वविद्यालय)

हिशाम अल-मुबैद (ह्यूस्टन विश्वविद्यालय-क्लियर लेक)

प्रतिमा अक्कुनूर (एरिज़ोना स्टेट यूनिवर्सिटी)

पॉल आमेर (डेलावेयर विश्वविद्यालय)

शमीउल आज़ोम (एरिज़ोना स्टेट यूनिवर्सिटी)

लिचुन बाओ (कैलिफोर्निया विश्वविद्यालय, इरविन)

पॉल बारफोर्ड (विस्कॉन्सिन विश्वविद्यालय)

बॉबी भट्टाचार्य (मैरीलैंड विश्वविद्यालय)

स्टीवन बेलोविन (कोलंबिया विश्वविद्यालय)

प्रवीण भागवत (विभु)

सुप्रतीक भट्टाचार्य (पूर्व में स्प्रिंट में)

अर्न्स्ट बायर्सैक (यूरेकॉम इंस्टीट्यूट)

शाहिद बोखारी (इंजीनियरिंग एवं प्रौद्योगिकी विश्वविद्यालय, लाहौर)

जीन बोलोट (टेक्नीकलर रिसर्च)

डैनियल ब्रुशटेन (पेन्सिलवेनिया विश्वविद्यालय के पूर्व छात्र)

केन कैल्वर्ट (केंटकी विश्वविद्यालय)

इवांड्रो कैंतु (फेडरल यूनिवर्सिटी ऑफ सांता कैटरीना)

जेफ़ केस (एसएनएमपी रिसर्च इंटरनेशनल)

जेफ़ चाल्टास (स्प्रिंट)

विंटन सेर्फ़ (गूगल)

ब्युंग क्यू चोई (मिशिगन टेक्नोलॉजिकल यूनिवर्सिटी)

ब्रैम कोहेन (बिटटोरेंट, इंक.)

कॉन्स्टेंटाइन कॉट्रास (पेस यूनिवर्सिटी)

जॉन डेगल (मिसिसिपी विश्वविद्यालय)

एडमंडो ए. डी सूजा ई सिल्वा (फेडरल यूनिवर्सिटी ऑफ़ रियो डी जनेरियो)

फिलिप डेक्यूटोस (यूरेकॉम इंस्टीट्यूट)

क्रिस्टोफ़ डिओट (टेक्नीकलर रिसर्च)

पृथुला धुंघेल (अकामाई)

डेबोरा एस्ट्रिन (कैलिफोर्निया विश्वविद्यालय, लॉस एंजिल्स)

मिचेलिस फालौत्सोस (रिवरसाइड में कैलिफोर्निया विश्वविद्यालय)

वू-ची फेंग (ओरेगन ग्रेजुएट इंस्टीट्यूट)

सैली फ्लॉयड (आईसीआईआर, कैलिफोर्निया विश्वविद्यालय, बर्कले)

पॉल फ्रांसिस (मैक्स प्लैंक इंस्टीट्यूट)

डेविड फुलगर (नेटफ्लिक्स)

लिक्सिन गाओ (मैसाचुसेट्स विश्वविद्यालय)

जे जे गार्सिया-लूना-ऐसेव्स (सांता क्रूज़ में कैलिफोर्निया विश्वविद्यालय)

मारियो गेर्ला (लॉस एंजिल्स में कैलिफोर्निया विश्वविद्यालय)

डेविड गुडमैन (NYU-पॉली)

यांग गुओ (अल्काटेल/ल्यूसेंट बेल लैब्स)

टिम ग्रिफिन (कैम्ब्रिज यूनिवर्सिटी)

मैक्स हैल्परिन (गुस्तावस एडोल्फस कॉलेज)

ब्रूस हार्वे (फ्लोरिडा ए एंड एम यूनिवर्सिटी, फ्लोरिडा स्टेट यूनिवर्सिटी)

कार्ल हॉसर (वाशिंगटन स्टेट यूनिवर्सिटी)

राचेल हेलर (जॉर्ज वाशिंगटन विश्वविद्यालय)

फिलिप होश्का (INRIA/W3C)

वेन ह्सिन (पार्क यूनिवर्सिटी)

अल्बर्ट हुआंग (पेन्सिलवेनिया विश्वविद्यालय के पूर्व छात्र)

चेंग हुआंग (माइक्रोसॉफ्ट रिसर्च)

एस्तेर ए. ह्यूजेस (वर्जीनिया कॉमनवेल्थ यूनिवर्सिटी)

वैन जैकबसन (ज़ेरॉक्स PARC)

पिनाक जैन (पूर्व NYU-पॉली छात्र)

जोबिन जेम्स (रिवरसाइड में कैलिफोर्निया विश्वविद्यालय)

सुगिह जामिन (मिशिगन विश्वविद्यालय)

शिवकुमार कल्याणरमन (आईबीएम रिसर्च, भारत)

जूसी कंगाशारजू (हेलसिंकी विश्वविद्यालय)

स्नेहा कसेरा (यूटा विश्वविद्यालय)

परवीज़ केरमानी (पूर्व में आईबीएम रिसर्च)

ह्योजिन किम (पेन्सिलवेनिया विश्वविद्यालय की पूर्व छात्रा)

लियोनार्ड क्लेनरॉक (लॉस एंजिल्स में कैलिफोर्निया विश्वविद्यालय)

डेविड कोट्ज़ (डार्टमाउथ कॉलेज)

बेशान कुलपाला (एरिज़ोना स्टेट यूनिवर्सिटी)

राकेश कुमार (ब्लूमबर्ग)

मिगुएल ए. लैब्राडोर (दक्षिण फ्लोरिडा विश्वविद्यालय)

साइमन लैम (टेक्सास विश्वविद्यालय)

स्टीव लाई (ओहियो स्टेट यूनिवर्सिटी)

टॉम लापोर्टा (पेन स्टेट यूनिवर्सिटी)

टिम-बर्नर्स ली (वर्ल्ड वाइड वेब कंसोर्टियम)

अरनॉड लेगौट (INRIA)

ली लिटनर (ड्रेक्सेल विश्वविद्यालय)

ब्रायन लेविन (मैसाचुसेट्स विश्वविद्यालय)

चुनचुन ली (पूर्व NYU-पॉली छात्र)

योंग लियू (एनवाईयू-पॉली)

विलियम लियांग (पेन्सिलवेनिया विश्वविद्यालय के पूर्व छात्र)

विलिस मार्टी (टेक्सास ए एंड एम यूनिवर्सिटी)

निक मैककेवन (स्टैनफोर्ड विश्वविद्यालय)

जोश मैकिन्ज़ी (पार्क यूनिवर्सिटी)

दीप मेधी (मिसौरी विश्वविद्यालय, कैनसस सिटी)

बॉब मेटकाफ (इंटरनेशनल डेटा ग्रुप)

सू मून (KAIST)

जेनी मोयेर (कॉमकास्ट)

एरिच नहूम (आईबीएम रिसर्च)

क्रिस्टोस पापाडोपोलोस (कोलोराडो स्टेट यूनिवर्सिटी)

क्रेग पार्ट्रिज (बीबीएन टेक्नोलॉजीज)

राडिया पर्लमैन (इंटेल)

जीतेन्द्र पाध्ये (माइक्रोसॉफ्ट रिसर्च)

वर्न पैक्सन (बर्कले में कैलिफोर्निया विश्वविद्यालय)

केविन फिलिप्स (स्प्रिंट)

जॉर्ज पॉलीज़ोस (एथेंस यूनिवर्सिटी ऑफ़ इकोनॉमिक्स एंड बिज़नेस)

श्रीराम राजगोपालन (एरिज़ोना स्टेट यूनिवर्सिटी)

रामचंद्रन रामजी (माइक्रोसॉफ्ट रिसर्च)

केन रीक (रोचेस्टर इंस्टीट्यूट ऑफ टेक्नोलॉजी)

मार्टिन रीसलीन (एरिज़ोना स्टेट यूनिवर्सिटी)

जेनिफर रेक्सफोर्ड (प्रिंसटन विश्वविद्यालय)

लियोन रेजनिक (रोचेस्टर इंस्टीट्यूट ऑफ टेक्नोलॉजी)

पाब्लो रोड्रिगेज़ (टेलीफ़ोनिका)

सुमित रॉय (वाशिंगटन विश्वविद्यालय)

डैन रूबेनस्टीन (कोलंबिया विश्वविद्यालय)

एवी रुबिन (जॉन्स हॉपकिन्स विश्वविद्यालय)

डगलस सैलेन (जॉन जे कॉलेज)

डेस्पिना सैपरिला (सिस्को सिस्टम्स)

जॉन शैन्ज़ (कॉमकास्ट)

हेनिंग शुल्ज़्रिने (कोलंबिया विश्वविद्यालय)

मिशा श्वार्टज़ (कोलंबिया विश्वविद्यालय)

आर्दश सेठी (डेलावेयर विश्वविद्यालय)

हरीश सेतु (ड्रेक्सल विश्वविद्यालय)

के. सैम शानमुगन (कैनसस विश्वविद्यालय)

प्रशांत शेनॉय (मैसाचुसेट्स विश्वविद्यालय)

क्ले शील्ड्स (जॉर्जटाउन विश्वविद्यालय)

सुबिन श्रेस्ट्रा (पेन्सिलवेनिया विश्वविद्यालय)

बोजी शू (पूर्व एनवाईयू-पॉली छात्र)

मिहैल एल. सिचितिउ (एनसी स्टेट यूनिवर्सिटी)

पीटर स्टीनकिस्टे (कार्नेगी मेलन विश्वविद्यालय)

तात्सुया सुदा (कैलिफोर्निया विश्वविद्यालय, इरविन)

किन सन टैम (अल्बानी में न्यूयॉर्क स्टेट यूनिवर्सिटी)

डॉन टॉस्ले (मैसाचुसेट्स विश्वविद्यालय)

डेविड टर्नर (कैलिफोर्निया स्टेट यूनिवर्सिटी, सैन बर्नार्डिनो)

नितिन वैद्य (इलिनोइस विश्वविद्यालय)

मिशेल वीगल (क्लेम्सन विश्वविद्यालय)

डेविड वेदरॉल (वाशिंगटन विश्वविद्यालय)

इरा विंस्टन (पेन्सिलवेनिया विश्वविद्यालय)

डि वू (सन यात-सेन विश्वविद्यालय)

शर्ली व्यान (NYU-पॉली)

राज यावतकर (इंटेल)

येचियाम येमिनी (कोलंबिया विश्वविद्यालय)

डियान यू (NYU शंघाई)

मिंग यू (स्टेट यूनिवर्सिटी ऑफ न्यूयॉर्क, बिंगहैमटन)

एलेन ज़ेगुरा (जॉर्जिया इंस्टीट्यूट ऑफ टेक्नोलॉजी)

होंगगैंग झांग (सफ़ोल्क विश्वविद्यालय)

हुई झांग (कार्नेगी मेलन विश्वविद्यालय)

लिक्सिया झांग (लॉस एंजिल्स में कैलिफोर्निया विश्वविद्यालय)

मेंग झांग (पूर्व NYU-पॉली छात्र)

शुचुन झांग (पेन्सिलवेनिया विश्वविद्यालय के पूर्व छात्र)

ज़ियाओडोंग झांग (ओहियो स्टेट यूनिवर्सिटी)

ज़िली झांग (मिनेसोटा विश्वविद्यालय)

फिल ज़िमरमैन (स्वतंत्र सलाहकार)

माइक जिंक (मैसाचुसेट्स विश्वविद्यालय)

क्लिफ सी. ज़ू (सेंट्रल फ्लोरिडा विश्वविद्यालय)

हम पूरी पियर्सन टीम को भी धन्यवाद देना चाहते हैं - खास तौर पर मैट गोल्डस्टीन और जोआन मैनिंग को - जिन्होंने इस सातवें संस्करण पर बिल्कुल शानदार काम किया है (और जिन्होंने दो बहुत ही बारीक लेखकों को संभाला है जो जन्मजात रूप से समयसीमा को पूरा करने में असमर्थ लगते हैं!)। हमारे कलाकारों, जेनेट थ्योरर और पैट्रिस रॉसी कैल्किन को भी इस और हमारी किताब के पिछले संस्करणों में सुंदर आकृतियों पर उनके काम के लिए धन्यवाद, और इस संस्करण पर उनके शानदार उत्पादन कार्य के लिए सेनवेओ में केटी ओस्टलर और उनकी टीम को भी धन्यवाद। अंत में, एडिसन-वेस्ले में हमारे पिछले दो संपादकों - माइकल हिर्श और सुसान हार्टमैन को सबसे खास धन्यवाद। उनके सुंदर प्रबंधन, निरंतर प्रोत्साहन, लगभग असीम धैर्य, अच्छे हास्य और दृढ़ता के बिना यह किताब वैसी नहीं होती जैसी यह है (और शायद बिल्कुल भी नहीं होती)।

विषयसूची

Machine Translated by Google

अध्याय 1 कंप्यूटर नेटवर्क और इंटरनेट

आज का इंटरनेट यकीनन मानव जाति द्वारा बनाया गया अब तक का सबसे बड़ा इंजीनियर्ड सिस्टम है, जिसमें करोड़ों कंप्यूटर, संचार लिंक और स्विच जुड़े हुए हैं; अरबों उपयोगकर्ता लैपटॉप, टैबलेट और स्मार्टफोन के माध्यम से जुड़ते हैं; और गेम कंसोल, निगरानी प्रणाली, घड़ियाँ, चश्मे, थर्मोस्टैट, बॉडी स्केल और कारों सहित कई नई इंटरनेट से जुड़ी "चीजें" हैं। यह देखते हुए कि इंटरनेट इतना बड़ा है और इसमें इतने सारे विविध घटक और उपयोग हैं, क्या यह समझने की कोई उम्मीद है कि यह कैसे काम करता है? क्या ऐसे मार्गदर्शक सिद्धांत और संरचना हैं जो इतनी बड़ी और जटिल प्रणाली को समझने के लिए आधार प्रदान कर सकते हैं? और यदि ऐसा है, तो क्या यह संभव है कि यह वास्तव में हो

कंप्यूटर नेटवर्क के बारे में सीखना कितना रोचक और मजेदार हो सकता है? सौभाग्य से, इन सभी सवालों का जवाब एक जोरदार हाँ है! वास्तव में, इस पुस्तक में हमारा उद्देश्य आपको कंप्यूटर नेटवर्किंग के गतिशील क्षेत्र का एक आधुनिक परिचय प्रदान करना है, जिससे आपको न केवल आज के नेटवर्क, बल्कि कल के नेटवर्क को समझने के लिए आवश्यक सिद्धांत और व्यावहारिक जानकारी मिल सके।

यह पहला अध्याय कंप्यूटर नेटवर्किंग और इंटरनेट का एक व्यापक अवलोकन प्रस्तुत करता है। यहाँ हमारा लक्ष्य एक व्यापक चित्र प्रस्तुत करना और इस पुस्तक के बाकी हिस्सों के लिए संदर्भ निर्धारित करना है, ताकि पेड़ों के माध्यम से जंगल को देखा जा सके।
हम इस परिचयात्मक अध्याय में बहुत कुछ कवर करेंगे और कंप्यूटर नेटवर्क के बहुत से भागों पर चर्चा करेंगे, वह भी बड़े परिदृश्य से नजर हटाए बिना।

इस अध्याय में हम कंप्यूटर नेटवर्क के बारे में अपने अवलोकन को इस प्रकार से संरचित करेंगे। कुछ बुनियादी शब्दावली और अवधारणाओं को पेश करने के बाद, हम सबसे पहले बुनियादी हार्डवेयर और सॉफ़्टवेयर घटकों की जाँच करेंगे जो एक नेटवर्क बनाते हैं। हम नेटवर्क के किनारे से शुरू करेंगे और नेटवर्क में चल रहे अंतिम सिस्टम और नेटवर्क अनुप्रयोगों को देखेंगे। फिर हम कंप्यूटर नेटवर्क के मूल का पता लगाएंगे, डेटा को ट्रांसपोर्ट करने वाले लिंक और स्विच की जाँच करेंगे, साथ ही एक्सेस नेटवर्क और भौतिक मीडिया जो अंतिम सिस्टम को नेटवर्क कोर से जोड़ते हैं। हम सीखेंगे कि इंटरनेट नेटवर्क का एक नेटवर्क है, और हम सीखेंगे

जानें कि ये नेटवर्क एक दूसरे से कैसे जुड़ते हैं।

कंप्यूटर नेटवर्क के किनारे और कोर के इस अवलोकन को पूरा करने के बाद, हम इस अध्याय के दूसरे भाग में व्यापक और अधिक अमूर्त दृष्टिकोण लेंगे। हम कंप्यूटर नेटवर्क में डेटा की देरी, हानि और थ्रूपुट की जांच करेंगे और एंड-टू-एंड थ्रूपुट और देरी के लिए सरल मात्रात्मक मॉडल प्रदान करेंगे: मॉडल जो ट्रांसमिशन, प्रसार और कतार में देरी को ध्यान में रखते हैं। फिर हम कंप्यूटर नेटवर्किंग में कुछ प्रमुख वास्तुशिल्प सिद्धांतों, अर्थात् प्रोटोकॉल लेयरिंग और सेवा मॉडल का परिचय देंगे।

हम यह भी जानेंगे कि कंप्यूटर नेटवर्क कई अलग-अलग प्रकार के हमलों के प्रति संवेदनशील होते हैं; हम सर्वेक्षण करेंगे

इनमें से कुछ हमलों पर चर्चा करें और विचार करें कि कंप्यूटर नेटवर्क को कैसे अधिक सुरक्षित बनाया जा सकता है। अंत में, हम कंप्यूटर नेटवर्किंग के संक्षिप्त इतिहास के साथ इस अध्याय को समाप्त करेंगे।

1.1 इंटरनेट क्या है?

इस पुस्तक में, हम सार्वजनिक इंटरनेट, एक विशिष्ट कंप्यूटर नेटवर्क, को हमारे मुख्य वाहन के रूप में उपयोग करेंगे।

कंप्यूटर नेटवर्क और उनके प्रोटोकॉल पर चर्चा करना। लेकिन इंटरनेट क्या है ? इस प्रश्न का उत्तर देने के दो तरीके हैं। सबसे पहले, हम इंटरनेट के मूल तत्वों का वर्णन कर सकते हैं, यानी, इंटरनेट को बनाने वाले बुनियादी हार्डवेयर और सॉफ़्टवेयर घटक। दूसरा, हम इंटरनेट को नेटवर्किंग इंफ्रास्ट्रक्चर के संदर्भ में वर्णित कर सकते हैं जो वितरित अनुप्रयोगों को सेवाएँ प्रदान करता है। आइए शुरू करते हैं

हमारी चर्चा को स्पष्ट करने के लिए चित्र 1.1 का उपयोग करते हुए, मूल विवरण को स्पष्ट करें ।

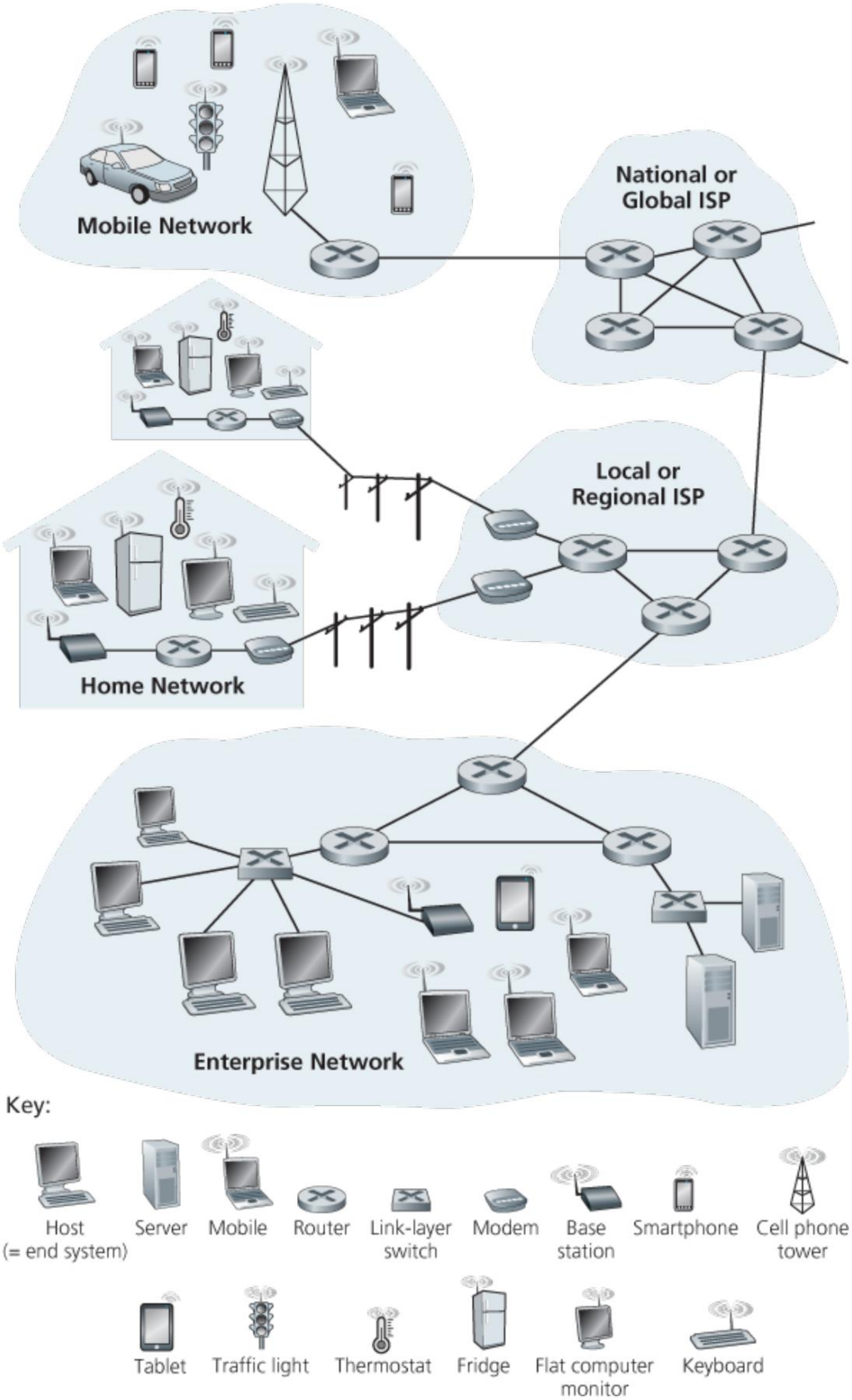
1.1.1 विस्तृत विवरण

इंटरनेट एक कंप्यूटर नेटवर्क है जो दुनिया भर में अरबों कंप्यूटिंग डिवाइस को आपस में जोड़ता है। बहुत समय पहले, ये कंप्यूटिंग डिवाइस मुख्य रूप से पारंपरिक डेस्कटॉप पीसी, लिनक्स वर्कस्टेशन और तथाकथित सर्वर थे जो वेब पेज और ई-मेल संदेशों जैसी जानकारी संग्रहीत और संचारित करते थे। हालाँकि, लैपटॉप, स्मार्टफ़ोन, टैबलेट, टीवी, गेमिंग कंसोल, थर्मोस्टैट, होम सिक्योरिटी सिस्टम, होम अप्लायंस, घड़ियाँ, आई ग्लास जैसी गैर-पारंपरिक इंटरनेट “चीजें” तेजी से लोकप्रिय हो रही हैं।

कारें, यातायात नियंत्रण प्रणाली और बहुत कुछ इंटरनेट से जोड़ा जा रहा है। वास्तव में, कंप्यूटर नेटवर्क **शब्द** थोड़ा पुराना लगने लगा है, क्योंकि कई गैर-पारंपरिक डिवाइस इंटरनेट से जुड़े हुए हैं। इंटरनेट शब्दावली में, इन सभी डिवाइस को होस्ट या एंड सिस्टम कहा जाता है। कुछ लोग

अनुमान है कि 2015 में लगभग 5 बिलियन डिवाइस इंटरनेट से जुड़े थे, और यह संख्या 2015 में 2 बिलियन से अधिक हो जाएगी।

2020 तक 25 बिलियन तक पहुँचने का लक्ष्य [गार्टनर 2014]। अनुमान है कि 2015 में दुनिया भर में 3.2 बिलियन से ज़्यादा इंटरनेट उपयोगकर्ता थे, जो दुनिया की आबादी का लगभग 40% है [आईटीयू 2015]।



चित्र 1.1 इंटरनेट के कुछ अंश

अंतिम प्रणालियाँ संचार लिंक और पैकेट स्विच के नेटवर्क द्वारा एक साथ जुड़ी हुई हैं ।

हम अनुभाग 1.2 में देखेंगे कि संचार लिंक कई प्रकार के होते हैं, जो निम्न से बने होते हैं:

विभिन्न प्रकार के भौतिक मीडिया, जिनमें समाक्षीय केबल, तांबे के तार, ऑप्टिकल फाइबर और रेडियो स्पेक्ट्रम शामिल हैं।
विभिन्न लिंक अलग-अलग दरों पर डेटा संचारित कर सकते हैं, एक लिंक की संचरण दर **को मापा जाता है**
बिट्स/सेकंड। जब एक एंड सिस्टम के पास दूसरे एंड सिस्टम को भेजने के लिए डेटा होता है, तो भेजने वाला एंड सिस्टम डेटा को सेगमेंट करता है और प्रत्येक सेगमेंट में हेडर बाइट्स जोड़ता है। परिणामी सूचना पैकेज, जिन्हें **कंप्यूटर नेटवर्क की भाषा में** पैकेट **के रूप में जाना जाता है ,** फिर नेटवर्क के माध्यम से गंतव्य एंड सिस्टम को भेजे जाते हैं, जहाँ उन्हें मूल डेटा में फिर से जोड़ा जाता है।

पैकेट स्विच अपने इनकमिंग कम्युनिकेशन लिंक में से किसी एक पर आने वाले पैकेट को लेता है और उस पैकेट को अपने आउटगोइंग कम्युनिकेशन लिंक में से किसी एक पर फॉरवर्ड करता है। पैकेट स्विच कई आकार और फ्लेवर में आते हैं, लेकिन आज के इंटरनेट में दो सबसे प्रमुख प्रकार राउटर और लिंक-लेयर स्विच हैं। दोनों प्रकार के स्विच पैकेट को उनके अंतिम गंतव्य की ओर अग्रेषित करते हैं। लिंक-लेयर स्विच आमतौर पर एक्सेस नेटवर्क में उपयोग किए जाते हैं, जबकि राउटर आमतौर पर नेटवर्क कोर में उपयोग किए जाते हैं। संचार लिंक और पैकेट स्विच का अनुक्रम जो पैकेट द्वारा भेजने वाले एंड सिस्टम से प्राप्त करने वाले एंड सिस्टम तक जाता है, उसे नेटवर्क के माध्यम से एक मार्ग या पथ के **रूप में जाना जाता है** । सिस्को का अनुमान है कि वार्षिक वैश्विक आईपी ट्रैफ़िक गुजरेगा

2016 के अंत तक ज़ेटाबाइट (10^{21} बाइट्स) की सीमा, तथा 2019 तक 2 ज़ेटाबाइट प्रति वर्ष तक पहुँच जाएगी

[सिस्को वीएनआई 2015].

पैकेट-स्विच्ड नेटवर्क (जो पैकेट का परिवहन करते हैं) कई मायनों में राजमार्गों, सड़कों और चौराहों (जो वाहनों का परिवहन करते हैं) के परिवहन नेटवर्क के समान हैं। उदाहरण के लिए, एक कारखाने पर विचार करें जिसे हजारों किलोमीटर दूर स्थित किसी गंतव्य गोदाम में बड़ी मात्रा में माल ले जाने की आवश्यकता है। कारखाने में, माल को विभाजित किया जाता है और ट्रकों के बेड़े में लोड किया जाता है। फिर प्रत्येक ट्रक स्वतंत्र रूप से राजमार्गों, सड़कों और चौराहों के नेटवर्क से गंतव्य गोदाम तक जाता है। गंतव्य गोदाम में, माल को उतार दिया जाता है और उसी शिपमेंट से आने वाले बाकी माल के साथ समूहीकृत किया जाता है। इस प्रकार, कई मायनों में, पैकेट ट्रकों के अनुरूप होते हैं, संचार लिंक राजमार्गों और सड़कों के अनुरूप होते हैं, पैकेट स्विच चौराहों के अनुरूप होते हैं, और अंत प्रणाली इमारतों के अनुरूप होती हैं। जिस तरह एक ट्रक परिवहन नेटवर्क के माध्यम से एक रास्ता लेता है, उसी तरह एक पैकेट कंप्यूटर नेटवर्क के माध्यम से एक रास्ता लेता है।

एंड सिस्टम इंटरनेट सेवा प्रदाताओं (आईएसपी) के माध्यम से इंटरनेट का उपयोग करते हैं , जिसमें स्थानीय केबल या टेलीफोन कंपनियों जैसे आवासीय आईएसपी शामिल हैं; कॉर्पोरेट आईएसपी; विश्वविद्यालय आईएसपी; आईएसपी जो हवाई अड्डों, होटलों, कॉफी की दुकानों और अन्य सार्वजनिक स्थानों में वाईफाई का उपयोग प्रदान करते हैं; और सेलुलर डेटा आईएसपी, हमारे स्मार्टफोन और अन्य उपकरणों तक मोबाइल पहुंच प्रदान करते हैं। प्रत्येक आईएसपी अपने आप में पैकेट स्विच और संचार लिंक का एक नेटवर्क है। आईएसपी एंड सिस्टम को विभिन्न प्रकार की नेटवर्क पहुंच प्रदान करते हैं, जिसमें केबल मॉडेम या डीएसएल जैसे आवासीय ब्रॉडबैंड एक्सेस, हाई-स्पीड लोकल एरिया नेटवर्क एक्सेस और मोबाइल वायरलेस एक्सेस शामिल हैं। आईएसपी कंटेंट प्रदाताओं को भी इंटरनेट एक्सेस प्रदान करते हैं,

ये निचले स्तर के ISP राष्ट्रीय और अंतर्राष्ट्रीय ऊपरी स्तर के ISP जैसे कि लेवल 3 कम्युनिकेशंस, AT&T, स्प्रिंट और NTT के माध्यम से आपस में जुड़े हुए हैं। एक ऊपरी स्तर के ISP में उच्च गति वाले राउटर होते हैं जो उच्च गति वाले फाइबर-ऑप्टिक लिंक से जुड़े होते हैं। प्रत्येक ISP नेटवर्क, चाहे ऊपरी स्तर का हो या निचला स्तर का, एक दूसरे से जुड़ा हुआ है।

स्वतंत्र रूप से प्रबंधित, आईपी प्रोटोकॉल चलाता है (नीचे देखें), और कुछ नामकरण और पते के अनुरूप है

हम अनुभाग 1.3 में आईएसपी और उनके अंतर्संबंधों की अधिक बारीकी से जांच करेंगे ।

इंटरनेट के एंड सिस्टम, पैकेट स्विच और अन्य हिस्से प्रोटोकॉल **चलाते हैं** जो इंटरनेट के भीतर सूचना भेजने और प्राप्त करने को नियंत्रित करते हैं। ट्रांसमिशन कंट्रोल प्रोटोकॉल (TCP) और इंटरनेट प्रोटोकॉल (IP) इंटरनेट में दो सबसे महत्वपूर्ण प्रोटोकॉल हैं। IP प्रोटोकॉल राउटर और एंड सिस्टम के बीच भेजे और प्राप्त किए जाने वाले पैकेट के प्रारूप को निर्दिष्ट करता है। इंटरनेट के प्रमुख प्रोटोकॉल को सामूहिक रूप से TCP/IP के रूप में जाना जाता है। हम इस परिचयात्मक अध्याय में प्रोटोकॉल पर विचार करना शुरू करेंगे। लेकिन यह सिर्फ एक शुरुआत है - इस पुस्तक का अधिकांश भाग कंप्यूटर नेटवर्क से संबंधित है

प्रोटोकॉल!

इंटरनेट के लिए प्रोटोकॉल के महत्व को देखते हुए, यह महत्वपूर्ण है कि हर कोई इस बात पर सहमत हो कि प्रत्येक व्यक्ति क्या चाहता है। हर प्रोटोकॉल ऐसा करता है, ताकि लोग ऐसे सिस्टम और उत्पाद बना सकें जो आपस में एक दूसरे से जुड़ सकें। यहीं पर मानक लागू होते हैं। इंटरनेट मानकों का विकास इंटरनेट इंजीनियरिंग टास्क फोर्स द्वारा किया जाता है

(IETF) [IETF 2016]। IETF मानक दस्तावेजों को टिप्पणियों के लिए अनुरोध (RFC) कहा जाता है। RFC की शुरुआत नेटवर्क और प्रोटोकॉल डिज़ाइन को हल करने के लिए टिप्पणियों के लिए सामान्य अनुरोध (इसलिए नाम) के रूप में हुई थी

इंटरनेट के अग्रदूतों के सामने आने वाली समस्याएं [ऑलमैन 2011]। RFC काफी तकनीकी और विस्तृत होते हैं। वे TCP, IP, HTTP (वेब के लिए) और SMTP (ई-मेल के लिए) जैसे प्रोटोकॉल को परिभाषित करते हैं। वर्तमान में 7,000 से अधिक RFC हैं। अन्य निकाय भी नेटवर्क घटकों के लिए मानक निर्दिष्ट करते हैं, जिनमें से अधिकांश

खास तौर पर नेटवर्क लिंक के लिए। उदाहरण के लिए, IEEE 802 LAN/MAN मानक समिति [IEEE 802 2016] ईथरनेट और वायरलेस WiFi मानकों को निर्दिष्ट करती है।

1.1.2 सेवा विवरण

ऊपर की हमारी चर्चा ने इंटरनेट को बनाने वाले कई हिस्सों की पहचान की है। लेकिन हम यह भी जान सकते हैं

इंटरनेट को एक बिलकुल अलग कोण से वर्णित करें - अर्थात, एक ऐसे बुनियादी ढांचे के रूप में जो अनुप्रयोगों को सेवाएँ प्रदान करता है। ई-मेल और वेब सर्फिंग जैसे पारंपरिक अनुप्रयोगों के अलावा, इंटरनेट अनुप्रयोगों में मोबाइल स्मार्टफोन और टैबलेट अनुप्रयोग शामिल हैं, जिनमें इंटरनेट मैसेजिंग, मैपिंग शामिल हैं

वास्तविक समय की सड़क-यातायात जानकारी, क्लाउड से संगीत स्ट्रीमिंग, फिल्म और टेलीविजन स्ट्रीमिंग, ऑनलाइन सामाजिक नेटवर्क, वीडियो कॉन्फ्रेंसिंग, बहु-व्यक्ति गेम और स्थान-आधारित अनुशंसा के साथ

सिस्टम। अनुप्रयोगों को वितरित अनुप्रयोग कहा जाता है, क्योंकि उनमें कई अंत शामिल होते हैं

सिस्टम जो एक दूसरे के साथ डेटा का आदान-प्रदान करते हैं। महत्वपूर्ण बात यह है कि इंटरनेट एप्लीकेशन अंतिम सिस्टम पर चलते हैं—

वे नेटवर्क कोर में पैकेट स्विच में नहीं चलते हैं। हालाँकि पैकेट स्विच अंतिम सिस्टम के बीच डेटा के आदान-प्रदान की सुविधा प्रदान करते हैं, लेकिन वे उस एप्लिकेशन से संबंधित नहीं हैं जो स्रोत या डेटा का सिंक.

आइए थोड़ा और विस्तार से जानें कि एप्लीकेशन को सेवाएँ प्रदान करने वाले बुनियादी ढांचे से हमारा क्या अभिप्राय है। इस उद्देश्य के लिए, मान लीजिए कि आपके पास वितरित इंटरनेट एप्लीकेशन के लिए एक रोमांचक नया विचार है, जो मानवता को बहुत लाभ पहुँचा सकता है या जो आपको बस अमीर और प्रसिद्ध बना सकता है। आप कैसे आगे बढ़ सकते हैं

इस विचार को वास्तविक इंटरनेट एप्लिकेशन में बदलना? क्योंकि एप्लिकेशन एंड सिस्टम पर चलते हैं, इसलिए आपको ऐसे प्रोग्राम लिखने होंगे जो एंड सिस्टम पर चलते हों। उदाहरण के लिए, आप अपने प्रोग्राम जावा, सी या पायथन में लिख सकते हैं। अब, क्योंकि आप एक वितरित इंटरनेट एप्लिकेशन विकसित कर रहे हैं, इसलिए अलग-अलग एंड सिस्टम पर चलने वाले प्रोग्राम को एक-दूसरे को डेटा भेजने की आवश्यकता होगी। और यहाँ हम एक केंद्रीय मुद्दे पर पहुँचते हैं - जो इंटरनेट को एक प्लेटफ़ॉर्म के रूप में वर्णित करने के वैकल्पिक तरीके की ओर ले जाता है

अनुप्रयोग। एक छोर पर चलने वाला एक प्रोग्राम इंटरनेट को डेटा वितरित करने का निर्देश कैसे देता है
क्या कोई अन्य प्रोग्राम किसी अन्य सिस्टम पर चल रहा है?

इंटरनेट से जुड़े एंड सिस्टम एक सॉकेट इंटरफ़ेस **प्रदान करते हैं** जो निर्दिष्ट करता है कि एक एंड सिस्टम पर चलने वाला प्रोग्राम इंटरनेट इन्फ्रास्ट्रक्चर को दूसरे एंड सिस्टम पर चलने वाले किसी विशिष्ट गंतव्य प्रोग्राम को डेटा डिलीवर करने के लिए कैसे कहता है। यह इंटरनेट सॉकेट इंटरफ़ेस नियमों का एक सेट है जिसका पालन भेजने वाले प्रोग्राम को करना चाहिए ताकि इंटरनेट गंतव्य प्रोग्राम को डेटा डिलीवर कर सके। हम चर्चा करेंगे

अध्याय 2 में इंटरनेट सॉकेट इंटरफ़ेस के बारे में विस्तार से बताया गया है । अभी के लिए, आइए एक सरल सादृश्य लें, जिसका उपयोग हम इस पुस्तक में अक्सर करेंगे। मान लीजिए कि एलिस डाक सेवा का उपयोग करके बॉब को एक पत्र भेजना चाहती है।
बेशक, ऐलिस सिर्फ़ पत्र (डेटा) लिखकर उसे अपनी खिड़की से बाहर नहीं फेंक सकती। इसके बजाय, डाक सेवा चाहती है कि ऐलिस पत्र को एक लिफ़ाफ़े में रखे; लिफ़ाफ़े के बीच में बॉब का पूरा नाम, पता और ज़िप कोड लिखें; लिफ़ाफ़े को सील करें; लिफ़ाफ़े के ऊपरी-दाएँ कोने में एक मोहर लगाएँ; और अंत में, लिफ़ाफ़े को आधिकारिक डाक सेवा मेलबॉक्स में डालें। इस प्रकार, डाक सेवा का अपना "डाक सेवा इंटरफ़ेस" या नियमों का सेट होता है, जिसका पालन ऐलिस को डाक सेवा द्वारा बॉब तक अपना पत्र पहुँचाने के लिए करना चाहिए। इसी तरह, इंटरनेट में एक सॉकेट इंटरफ़ेस होता है जिसका पालन डेटा भेजने वाले प्रोग्राम को करना चाहिए ताकि इंटरनेट डेटा को उस प्रोग्राम तक पहुँचा सके जो डेटा प्राप्त करेगा।

डाक सेवा, निश्चित रूप से, अपने ग्राहकों को एक से अधिक सेवाएँ प्रदान करती है। यह एक्सप्रेस डिलीवरी, रिसेप्शन कन्फर्मेशन, साधारण उपयोग और कई अन्य सेवाएँ प्रदान करती है। इसी तरह, इंटरनेट भी
अपने अनुप्रयोगों को कई सेवाएँ प्रदान करता है। जब आप एक इंटरनेट अनुप्रयोग विकसित करते हैं, तो आपको भी
अपने आवेदन के लिए इंटरनेट की सेवाओं में से एक चुनें। हम इंटरनेट की सेवाओं का वर्णन करेंगे
अध्याय दो।

हमने अभी इंटरनेट के दो विवरण दिए हैं; एक इसके हार्डवेयर और सॉफ़्टवेयर घटकों के संदर्भ में, दूसरा वितरित अनुप्रयोगों को सेवाएँ प्रदान करने के लिए एक बुनियादी ढाँचे के संदर्भ में। लेकिन शायद आप अभी भी भ्रमित हैं कि इंटरनेट क्या है। पैकेट स्विचिंग और TCP/IP क्या हैं? राउटर क्या हैं? इंटरनेट में किस तरह के संचार लिंक मौजूद हैं? वितरित अनुप्रयोग क्या है? थर्मोस्टेट या बॉडी स्केल को इंटरनेट से कैसे जोड़ा जा सकता है? अगर आपको अब यह सब समझ में नहीं आ रहा है, तो चिंता न करें - इस पुस्तक का उद्देश्य आपको इंटरनेट के मूल तत्वों और इसके काम करने के तरीके और कारणों को नियंत्रित करने वाले सिद्धांतों से परिचित कराना है। हम इन महत्वपूर्ण शब्दों और प्रश्नों को निम्नलिखित अनुभागों और अध्यायों में समझाएँगे।

1.1.3 प्रोटोकॉल क्या है?

अब जबकि हमें इंटरनेट के बारे में कुछ जानकारी मिल गई है, तो आइए इंटरनेट के एक अन्य महत्वपूर्ण शब्द पर विचार करें।

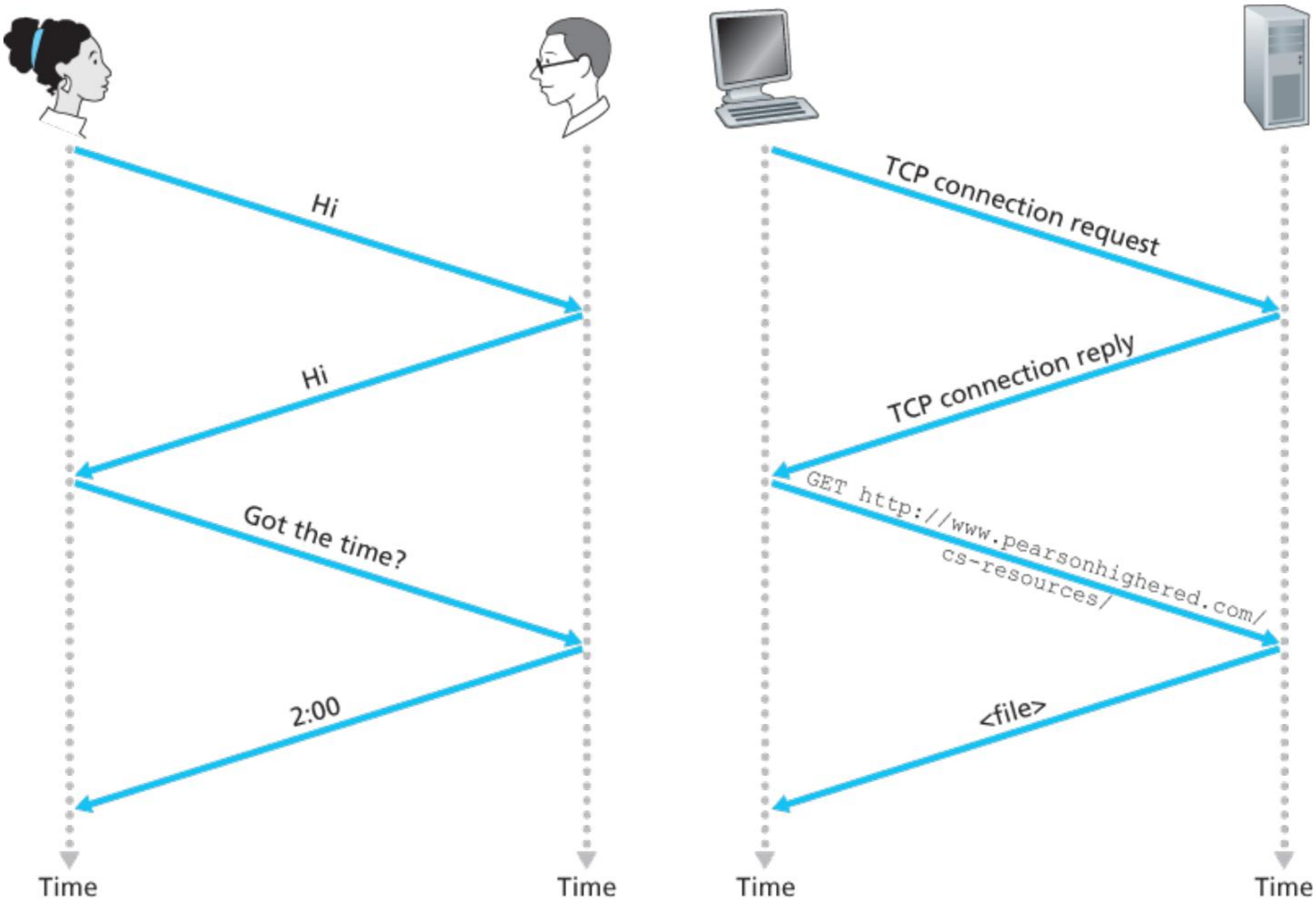
कंप्यूटर नेटवर्किंग: प्रोटोकॉल। प्रोटोकॉल क्या है? प्रोटोकॉल क्या करता है?

एक मानवीय सादृश्य

कंप्यूटर नेटवर्क प्रोटोकॉल की अवधारणा को समझना शायद सबसे आसान है, पहले कुछ मानवीय समानताओं पर विचार करके, क्योंकि हम मनुष्य हर समय प्रोटोकॉल निष्पादित करते हैं। विचार करें कि जब आप ऐसा करते हैं तो आप क्या करते हैं

किसी से दिन का समय पूछना चाहते हैं। एक सामान्य आदान-प्रदान चित्र 1.2 में दिखाया गया है। मानवीय प्रोटोकॉल (या कम से कम अच्छे शिष्टाचार) के अनुसार किसी व्यक्ति को किसी अन्य व्यक्ति के साथ संचार शुरू करने के लिए पहले अभिवादन (चित्र 1.2 में पहला "हाय") करना चाहिए। "हाय" के लिए सामान्य प्रतिक्रिया एक "हाय" संदेश है। निहित रूप से, व्यक्ति तब एक सौहार्दपूर्ण "हाय" प्रतिक्रिया को एक संकेत के रूप में लेता है कि वह आगे बढ़ सकता है और दिन का समय पूछ सकता है।

प्रारंभिक "हाय" के लिए एक अलग प्रतिक्रिया (जैसे "मुझे परेशान मत करो!" या "मैं अंग्रेजी नहीं बोलता," या कुछ अप्रकाशित उत्तर)



चित्र 1.2 एक मानव प्रोटोकॉल और एक कंप्यूटर नेटवर्क प्रोटोकॉल

संवाद करने में अनिच्छा या असमर्थता को इंगित करें। इस मामले में, मानव प्रोटोकॉल दिन का समय नहीं पूछना होगा। कभी-कभी किसी प्रश्न का कोई उत्तर नहीं मिलता है, ऐसे में आमतौर पर कोई व्यक्ति

उस व्यक्ति से समय पूछना छोड़ देता है। ध्यान दें कि हमारे मानव प्रोटोकॉल में, विशिष्ट संदेश हैं

हम जो संदेश भेजते हैं, और प्राप्त उत्तर संदेशों या अन्य घटनाओं (जैसे कि कुछ निश्चित समय के भीतर कोई उत्तर न मिलना) के जवाब में हम जो विशिष्ट कार्य करते हैं। स्पष्ट रूप से, प्रेषित और प्राप्त संदेश, और जब ये संदेश भेजे या प्राप्त किए जाते हैं या अन्य घटनाएँ होती हैं, तो की गई क्रियाएँ, मानव प्रोटोकॉल में केंद्रीय भूमिका निभाती हैं। यदि लोग अलग-अलग प्रोटोकॉल चलाते हैं (उदाहरण के लिए, यदि एक व्यक्ति के पास शिष्टाचार है लेकिन दूसरे के पास नहीं है, या यदि एक समय की अवधारणा को समझता है और दूसरा नहीं) तो प्रोटोकॉल आपस में नहीं जुड़ते हैं और कोई उपयोगी कार्य पूरा नहीं हो सकता है। नेटवर्किंग में भी यही सच है - किसी कार्य को पूरा करने के लिए एक ही प्रोटोकॉल चलाने वाली दो (या अधिक) संचार करने वाली संस्थाओं की आवश्यकता होती है।

आइए एक दूसरे मानवीय उदाहरण पर विचार करें। मान लीजिए कि आप कॉलेज की कक्षा में हैं (उदाहरण के लिए, कंप्यूटर नेटवर्किंग की कक्षा!)। शिक्षक प्रोटोकॉल के बारे में लगातार बोल रहा है और आप भ्रमित हैं। शिक्षक रुककर पूछता है, "क्या कोई प्रश्न है?" (एक संदेश जो उन सभी छात्रों को प्रेषित किया जाता है, और उन्हें प्राप्त होता है जो सो नहीं रहे हैं)। आप अपना हाथ उठाते हैं (शिक्षक को एक अंतर्निहित संदेश प्रेषित करते हुए)। आपका शिक्षक आपको मुस्कुराते हुए स्वीकार करता है, "हाँ . . ." कहता है (एक प्रेषित संदेश जो आपको अपने प्रश्न पूछने के लिए प्रोत्साहित करता है

प्रश्न—शिक्षकों को **प्रश्न पूछे जाने** पसंद होते हैं), और फिर आप अपना प्रश्न पूछते हैं (अर्थात, अपने शिक्षक को अपना संदेश प्रेषित करते हैं)। आपका शिक्षक आपका प्रश्न सुनता है (आपका प्रश्न संदेश प्राप्त करता है) और उत्तर देता है (आपको उत्तर प्रेषित करता है)। एक बार फिर, हम देखते हैं कि संदेशों का प्रसारण और प्राप्ति, और इन संदेशों को भेजे और प्राप्त किए जाने पर की जाने वाली पारंपरिक क्रियाओं का एक सेट, इस प्रश्न-उत्तर प्रोटोकॉल के केंद्र में है।

नेटवर्क प्रोटोकॉल

नेटवर्क प्रोटोकॉल मानव प्रोटोकॉल के समान ही होता है, सिवाय इसके कि इसमें संदेश और सूचनाओं का आदान-प्रदान करने वाली संस्थाएं एक-दूसरे से संवाद करती हैं। कार्रवाई करने वाले किसी डिवाइस के हार्डवेयर या सॉफ़्टवेयर घटक होते हैं (उदाहरण के लिए, कंप्यूटर, स्मार्टफ़ोन, टैबलेट, राउटर या अन्य नेटवर्क-सक्षम डिवाइस)। इंटरनेट पर सभी गतिविधियाँ जिसमें दो शामिल होते हैं
या अधिक संचार करने वाली दूरस्थ संस्थाओं को एक प्रोटोकॉल द्वारा नियंत्रित किया जाता है। उदाहरण के लिए, हार्डवेयर-क्रियान्वित दो भौतिक रूप से जुड़े कंप्यूटरों में प्रोटोकॉल दो नेटवर्क इंटरफ़ेस कार्ड के बीच "वायर" पर बिट्स के प्रवाह को नियंत्रित करते हैं; एंड सिस्टम में कंजेशन-कंट्रोल प्रोटोकॉल उस दर को नियंत्रित करते हैं जिस पर प्रेषक और रिसीवर के बीच पैकेट प्रसारित होते हैं; राउटर में प्रोटोकॉल स्रोत से गंतव्य तक पैकेट का मार्ग निर्धारित करते हैं। प्रोटोकॉल इंटरनेट में हर जगह चल रहे हैं, और परिणामस्वरूप इनमें से बहुत कुछ

पुस्तक कंप्यूटर नेटवर्क प्रोटोकॉल के बारे में है।

कंप्यूटर नेटवर्क प्रोटोकॉल के एक उदाहरण के रूप में, जिससे आप संभवतः परिचित हैं, विचार करें कि क्या होता है जब आप किसी वेब सर्वर से अनुरोध करते हैं, अर्थात, जब आप किसी वेब पेज का URL टाइप करते हैं
आपका वेब ब्राउज़र। परिदृश्य चित्र 1.2 के दाएँ आधे भाग में दर्शाया गया है। सबसे पहले, आपका कंप्यूटर वेब सर्वर को कनेक्शन अनुरोध संदेश भेजेगा और उत्तर की प्रतीक्षा करेगा। वेब सर्वर
अंततः आपका कनेक्शन अनुरोध संदेश प्राप्त होगा और कनेक्शन उत्तर संदेश वापस आएगा।
अब वेब दस्तावेज़ का अनुरोध करना ठीक है, फिर आपका कंप्यूटर उस वेब पेज का नाम GET संदेश में भेजता है जिसे वह उस वेब सर्वर से प्राप्त करना चाहता है। अंत में, वेब सर्वर आपके कंप्यूटर को वेब पेज (फ़ाइल) लौटा देता है।

उपरोक्त मानवीय और नेटवर्किंग उदाहरणों को देखते हुए, संदेशों का आदान-प्रदान और इन संदेशों को भेजे और प्राप्त किए जाने पर की जाने वाली कार्रवाइयां एक प्रोटोकॉल के प्रमुख परिभाषित तत्व हैं:

> एक प्रोटोकॉल दो या अधिक संचार करने वाली संस्थाओं के बीच आदान-प्रदान किए जाने वाले संदेशों के प्रारूप और क्रम को परिभाषित करता है, साथ ही संदेश के प्रेषण और/या प्राप्ति पर की जाने वाली कार्रवाइयों को भी परिभाषित करता है। या अन्य घटना.

इंटरनेट और सामान्य रूप से कंप्यूटर नेटवर्क, प्रोटोकॉल का व्यापक उपयोग करते हैं। विभिन्न संचार कार्यों को पूरा करने के लिए विभिन्न प्रोटोकॉल का उपयोग किया जाता है। जैसा कि आप इस पुस्तक को पढ़ते हैं, आप सीखेंगे कि कुछ प्रोटोकॉल सरल और सीधे हैं, जबकि अन्य जटिल और बौद्धिक रूप से गहरे हैं।

कंप्यूटर नेटवर्किंग के क्षेत्र में महारत हासिल करना, इसके क्या, क्यों और कैसे को समझने के बराबर है। नेटवर्किंग प्रोटोकॉल.

1.2 नेटवर्क एज

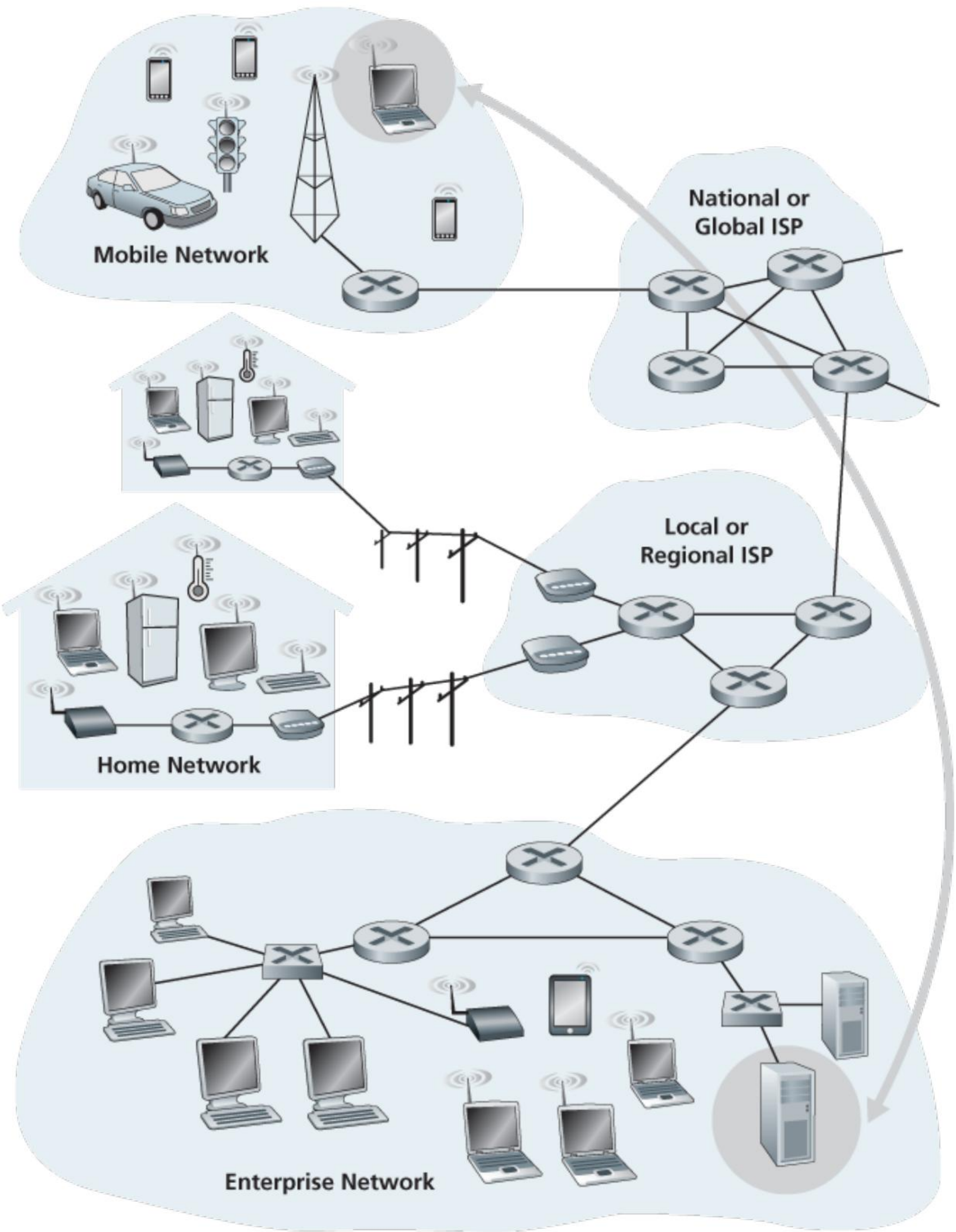
पिछले अनुभाग में हमने इंटरनेट और नेटवर्किंग प्रोटोकॉल का उच्च-स्तरीय अवलोकन प्रस्तुत किया था। अब हम कंप्यूटर नेटवर्क (और विशेष रूप से इंटरनेट) के घटकों के बारे में थोड़ा और गहराई से जानेंगे। हम इस अनुभाग में नेटवर्क के किनारे से शुरू करते हैं और उन घटकों को देखते हैं जिनसे हम सबसे अधिक परिचित हैं - अर्थात्, कंप्यूटर, स्मार्टफोन और अन्य डिवाइस जिन्हें हम दैनिक आधार पर उपयोग करते हैं। अगले अनुभाग में हम नेटवर्क किनारे से नेटवर्क कोर की ओर बढ़ेंगे और कंप्यूटर नेटवर्क में स्विचिंग और रूटिंग की जांच करेंगे।

पिछले भाग से याद करें कि कंप्यूटर नेटवर्किंग शब्दावली में, इंटरनेट से जुड़े कंप्यूटर और अन्य उपकरणों को अक्सर एंड सिस्टम के रूप में संदर्भित किया जाता है। उन्हें एंड सिस्टम के रूप में संदर्भित किया जाता है

क्योंकि वे इंटरनेट के किनारे पर बैठते हैं, जैसा कि चित्र 1.3 में दिखाया गया है। इंटरनेट के अंतिम सिस्टम में डेस्कटॉप कंप्यूटर (जैसे, डेस्कटॉप पीसी, मैक और लिनक्स बॉक्स), सर्वर (जैसे, वेब और ई-मेल सर्वर) और मोबाइल डिवाइस (जैसे, लैपटॉप, स्मार्टफोन और टैबलेट) शामिल हैं। इसके अलावा, गैर-पारंपरिक "चीजों" की बढ़ती संख्या को अंतिम सिस्टम के रूप में इंटरनेट से जोड़ा जा रहा है (केस हिस्ट्री फीचर देखें)।

अंतिम प्रणालियों को होस्ट भी कहा जाता है, क्योंकि वे अनुप्रयोग प्रोग्रामों को होस्ट करते हैं (अर्थात चलाते हैं), जैसे वेब ब्राउज़र प्रोग्राम, वेब सर्वर प्रोग्राम, ई-मेल क्लाइंट प्रोग्राम, या ई-मेल सर्वर प्रोग्राम।

इस पुस्तक में हम निम्नलिखित का प्रयोग करेंगे:



चित्र 1.3 अंत-प्रणाली अंतःक्रिया

> व्यक्ति वृत्त
>
> इंटरनेट ऑफ थिंग्स
>
> क्या आप ऐसी दुनिया की कल्पना कर सकते हैं जिसमें लगभग हर चीज़ वायरलेस तरीके से इंटरनेट से जुड़ी हो? एक ऐसी दुनिया जिसमें अधिकांश लोग, कार, साइकिल, चश्मे, घड़ियाँ, खिलौने, अस्पताल के उपकरण, घरेलू सेंसर, कक्षाएं, वीडियो निगरानी प्रणाली, वायुमंडलीय सेंसर, स्टोर-शेल्फ

क्या उत्पाद और पालतू जानवर आपस में जुड़े हुए हैं? इंटरनेट ऑफ थिंग्स (IoT) की यह दुनिया वास्तव में सिर्फ़ एक ही चीज़ हो सकती है कोने के आसपास।

कुछ अनुमानों के अनुसार, 2015 तक इंटरनेट से 5 बिलियन चीजें पहले से ही जुड़ी हुई हैं, और 2020 तक यह संख्या 25 बिलियन तक पहुँच सकती है [गार्टनर 2014]। इन चीज़ों में हमारे स्मार्टफ़ोन शामिल हैं, जो पहले से ही हमारे घरों, दफ़्तरों और कारों में हमारे साथ रहते हैं, हमारे भौगोलिक स्थानों और उपयोग के डेटा को हमारे ISP और इंटरनेट एप्लिकेशन को रिपोर्ट करते हैं। लेकिन हमारे स्मार्टफ़ोन के अलावा, कई तरह की गैर-पारंपरिक "चीज़ें" पहले से ही उत्पादों के रूप में उपलब्ध हैं। उदाहरण के लिए, इंटरनेट से जुड़े पहनने योग्य उपकरण हैं, जिनमें घड़ियाँ (Apple और कई अन्य से) और चश्मे शामिल हैं। उदाहरण के लिए, इंटरनेट से जुड़े चश्मे, हम जो कुछ भी देखते हैं उसे क्लाउड पर अपलोड कर सकते हैं, जिससे हम अपने दृश्य अनुभवों को वास्तविक समय में दुनिया भर के लोगों के साथ साझा कर सकते हैं। स्मार्ट होम के लिए पहले से ही इंटरनेट से जुड़ी चीज़ें उपलब्ध हैं, जिसमें इंटरनेट से जुड़े थर्मोस्टैट शामिल हैं जिन्हें हमारे स्मार्टफ़ोन से दूर से नियंत्रित किया जा सकता है, और इंटरनेट से जुड़े बॉडी स्केल, जो हमें अपने स्मार्टफ़ोन से अपने आहार की प्रगति की ग्राफ़िक रूप से समीक्षा करने में सक्षम बनाते हैं। इंटरनेट से जुड़े खिलौने हैं, जिनमें गुड़िया भी शामिल हैं जो बच्चे की बोली को पहचानती हैं और उसकी व्याख्या करती हैं और उचित तरीके से प्रतिक्रिया देती हैं।

IoT उपयोगकर्ताओं को संभावित रूप से क्रांतिकारी लाभ प्रदान करता है। लेकिन साथ ही साथ कुछ ऐसे भी लाभ हैं जो उपयोगकर्ताओं को क्रांतिकारी लाभ प्रदान करते हैं। सुरक्षा और गोपनीयता के लिए बहुत बड़ा जोखिम। उदाहरण के लिए, हमलावर इंटरनेट के ज़रिए हैक करने में सक्षम हो सकते हैं IoT डिवाइस में या IoT डिवाइस से डेटा एकत्र करने वाले सर्वर में। उदाहरण के लिए, एक हमलावर इंटरनेट से जुड़ी गुड़िया को हैक कर सकता है और सीधे बच्चे से बात कर सकता है; या हमलावर हैक कर सकता है पहनने योग्य उपकरणों से एकत्रित व्यक्तिगत स्वास्थ्य और गतिविधि की जानकारी को एक डेटाबेस में संग्रहीत किया जाता है। ये सुरक्षा और गोपनीयता संबंधी चिंताएँ उपभोक्ताओं के उस विश्वास को कमज़ोर कर सकती हैं जो प्रौद्योगिकियों को उनकी पूरी क्षमता तक पहुँचने के लिए आवश्यक है और इसका परिणाम कम व्यापक हो सकता है

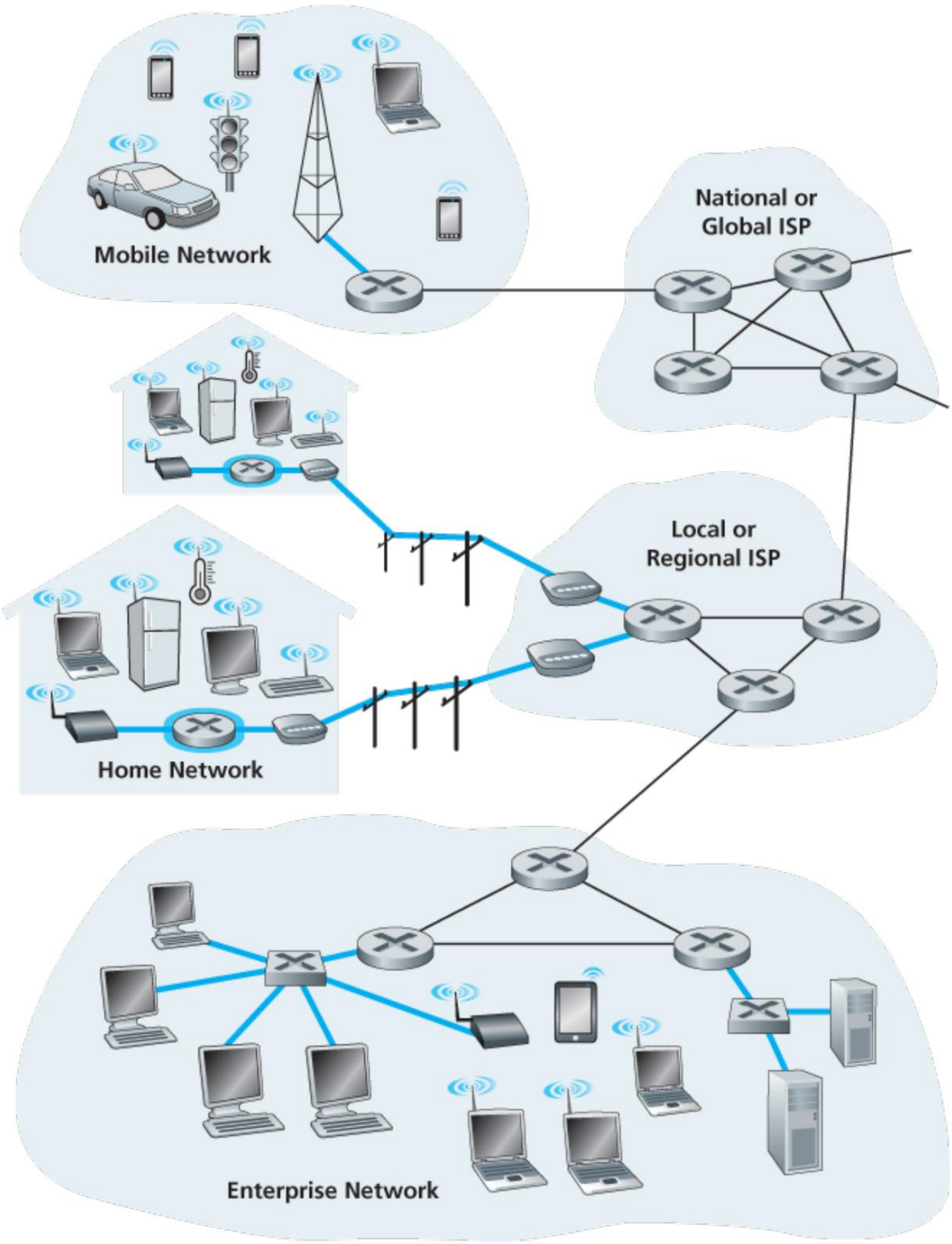
गोद लेना [एफटीसी 2015].

होस्ट और एंड सिस्टम को एक दूसरे के स्थान पर इस्तेमाल किया जाता है; यानी होस्ट = एंड सिस्टम। होस्ट को कभी-कभी दो श्रेणियों में विभाजित किया जाता है: क्लाइंट और सर्वर। अनौपचारिक रूप से, क्लाइंट डेस्कटॉप और मोबाइल पीसी, स्मार्टफोन आदि होते हैं, जबकि सर्वर अधिक शक्तिशाली मशीनें होती हैं जो वेब पेजों को संग्रहीत और वितरित करती हैं, वीडियो स्ट्रीम करती हैं, ईमेल रिले करती हैं, आदि। आज, अधिकांश सर्वर जिनसे हम खोज परिणाम, ईमेल, वेब पेज और वीडियो प्राप्त करते हैं , बड़े डेटा केंद्रों **में रहते हैं।** उदाहरण के लिए, Google के पास 50-100 डेटा केंद्र हैं, जिनमें लगभग 15 बड़े केंद्र शामिल हैं, जिनमें से प्रत्येक में 100,000 से अधिक सर्वर हैं।

1.2.1 नेटवर्क तक पहुंच

"नेटवर्क के किनारे" पर अनुप्रयोगों और अंतिम प्रणालियों पर विचार करने के बाद, आइए अब एक्सेस नेटवर्क पर विचार करें - वह नेटवर्क जो अंतिम सिस्टम को पहले राउटर (जिसे के रूप में भी जाना जाता है) से भौतिक रूप से जोड़ता है

“एज राउटर”) को अंतिम सिस्टम से किसी अन्य दूरस्थ अंतिम सिस्टम तक के पथ पर रखा जाता है। चित्र 1.4 कई प्रकार की पहुँच दिखाता है



चित्र 1.4 एक्सेस नेटवर्क

मोटी, छायांकित रेखाओं वाले नेटवर्क और वे सेटिंग्स (घर, उद्यम और विस्तृत क्षेत्र मोबाइल वायरलेस) जिनमें उनका उपयोग किया जाता है।

घरेलू पहुंच: डीएसएल, केबल, एफटीटीएच, डायल-अप और सैटेलाइट

विकसित देशों में 2014 तक 78 प्रतिशत से अधिक घरों में इंटरनेट की सुविधा है।

कोरिया, नीदरलैंड, फिनलैंड और स्वीडन 80 प्रतिशत से अधिक परिवारों के साथ अग्रणी हैं

इंटरनेट एक्सेस होना, लगभग सभी हाई-स्पीड ब्रॉडबैंड कनेक्शन के माध्यम से [आईटीयू 2015]। होम एक्सेस नेटवर्क के इस व्यापक उपयोग को देखते हुए आइए एक्सेस नेटवर्क के बारे में अपने अवलोकन की शुरुआत इस बात पर विचार करके करें

घर इंटरनेट से कैसे जुड़ते हैं।

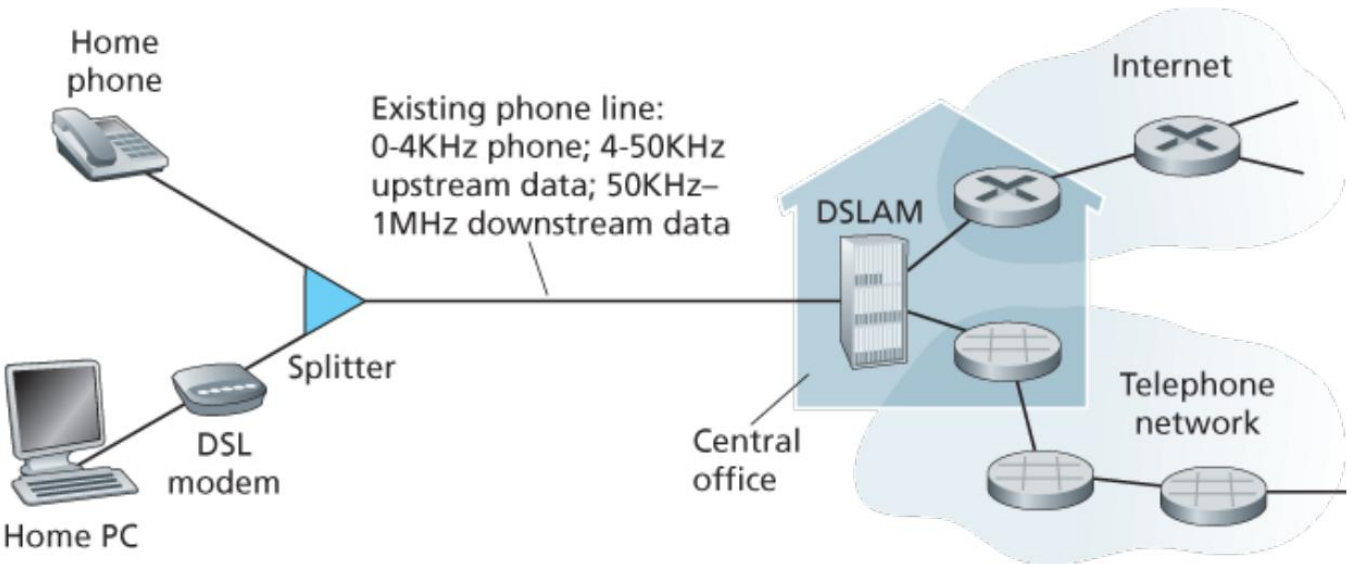
आज, ब्रॉडबैंड आवासीय पहुँच के दो सबसे प्रचलित प्रकार डिजिटल सब्सक्राइबर लाइन (DSL) और केबल हैं। एक निवास आमतौर पर उसी स्थानीय टेलीफोन कंपनी (टेल्को) से DSL इंटरनेट एक्सेस प्राप्त करता है जो उसे वायर्ड स्थानीय फ़ोन एक्सेस प्रदान करती है। इस प्रकार, जब DSL का उपयोग किया जाता है, तो ग्राहक की टेल्को भी उसकी होती है

आईएसपी। जैसा कि चित्र 1.5 में दिखाया गया है, प्रत्येक ग्राहक का DSL मॉडेम मौजूदा टेलीफोन लाइन (ट्विस्टेड-पेयर कॉपर वायर, जिसकी चर्चा हम अनुभाग 1.2.2 में करेंगे) का उपयोग टेलीकॉम कंपनी के स्थानीय केंद्रीय कार्यालय (सीओ) में स्थित डिजिटल सब्सक्राइबर लाइन एक्सेस मल्टीप्लेक्सर (डीएसएलएएम) के साथ डेटा का आदान-प्रदान करने के लिए करता है। घर का DSL मॉडेम डिजिटल डेटा लेता है और इसे सीओ को टेलीफोन तारों के माध्यम से संचारित करने के लिए उच्च-आवृत्ति टोन में अनुवाद करता है; ऐसे कई घरों से एनालॉग सिग्नल को डीएसएलएएम पर वापस डिजिटल प्रारूप में अनुवादित किया जाता है।

आवासीय टेलीफोन लाइन डेटा और पारंपरिक टेलीफोन सिग्नल दोनों को एक साथ ले जाती है, जिससे विभिन्न आवृत्तियों पर एनकोड किए गए हैं:

- 50 kHz से 1 MHz बैंड में एक उच्च गति वाला डाउनस्ट्रीम चैनल
- 4 kHz से 50 kHz बैंड में एक मध्यम गति अपस्ट्रीम चैनल
- 0 से 4 kHz बैंड में एक साधारण दो-तरफ़ा टेलीफ़ोन चैनल

इस दृष्टिकोण से एकल DSL लिंक ऐसा प्रतीत होता है मानो तीन अलग-अलग लिंक हों, जिससे एक टेलीफोन कॉल और एक इंटरनेट कनेक्शन एक ही समय में DSL लिंक को साझा कर सकें।



चित्र 1.5 डीएसएल इंटरनेट एक्सेस

(हम अनुभाग 1.3.1 में आवृत्ति-विभाजन बहुसंकेतन की इस तकनीक का वर्णन करेंगे ।) ग्राहक की ओर, एक स्प्लिटर घर में आने वाले डेटा और टेलीफोन संकेतों को अलग करता है और डेटा सिग्नल को आगे भेजता है।

डीएसएल मॉडेम। टेल्को की तरफ, सीओ में, डीएसएलएएम डेटा और फोन सिग्नल को अलग करता है और डेटा को इंटरनेट में भेजता है। सैकड़ों या हज़ारों घर एक ही डीएसएलएएम से जुड़ते हैं

[डिस्चिंगर 2007].

डीएसएल मानक कई संचरण दरों को परिभाषित करते हैं, जिनमें 12 एमबीपीएस डाउनस्ट्रीम और 1.8 एमबीपीएस शामिल हैं

अपस्ट्रीम [आईटीयू 1999], और 55 एमबीपीएस डाउनस्ट्रीम और 15 एमबीपीएस अपस्ट्रीम [आईटीयू 2006]। क्योंकि डाउनस्ट्रीम और अपस्ट्रीम दरें अलग-अलग हैं, इसलिए एक्सेस को असममित कहा जाता है। वास्तविक

प्राप्त डाउनस्ट्रीम और अपस्ट्रीम ट्रांसमिशन दरें ऊपर उल्लिखित दरों से कम हो सकती हैं, क्योंकि

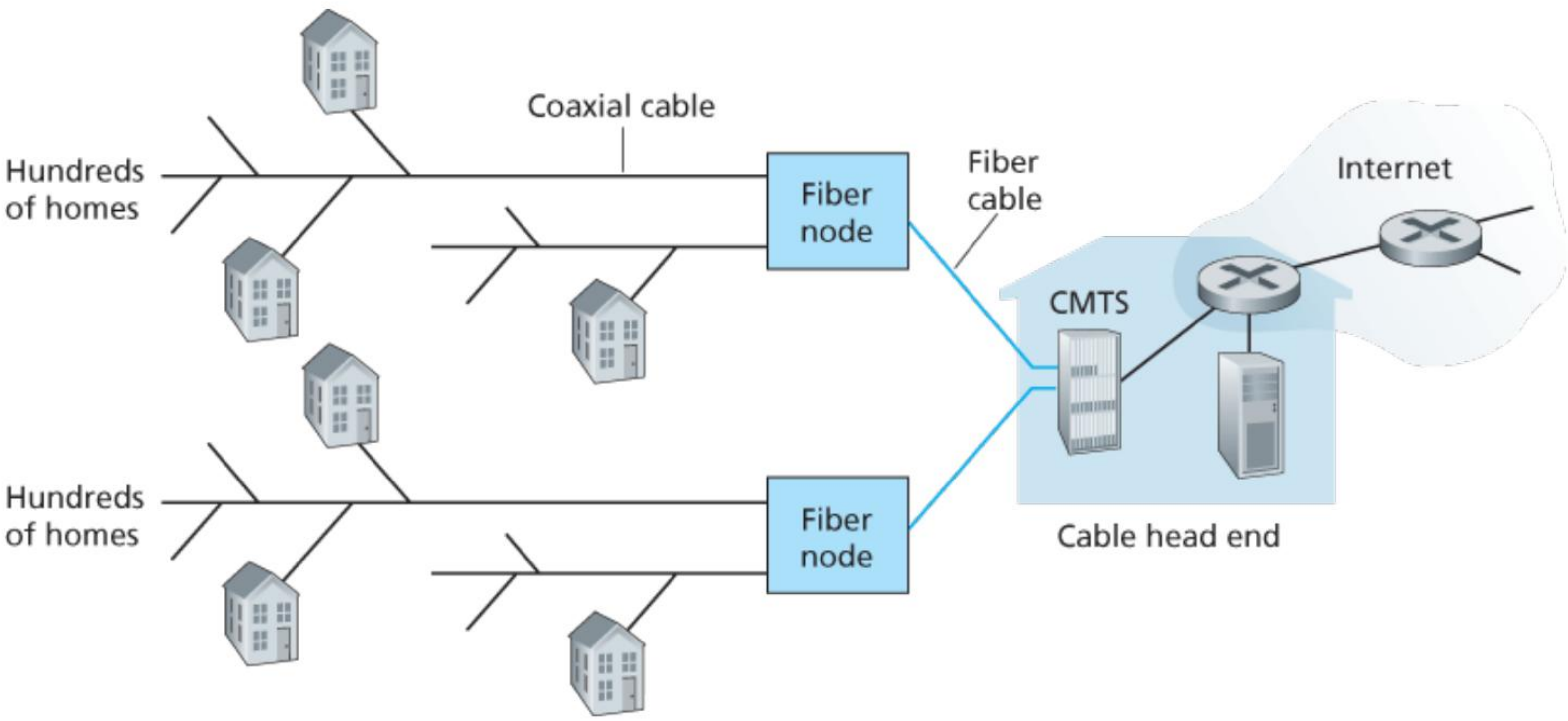
डीएसएल प्रदाता जानबूझकर आवासीय दर को सीमित कर सकता है जब स्तरीय सेवा (विभिन्न दरें, विभिन्न कीमतों पर उपलब्ध) की पेशकश की जाती है। अधिकतम दर घर और सीओ के बीच की दूरी, ट्विस्टेड-पेयर लाइन के गेज और विद्युत हस्तक्षेप की डिग्री द्वारा भी सीमित होती है। इंजीनियरों ने घर और सीओ के बीच छोटी दूरी के लिए डीएसएल को स्पष्ट रूप से डिजाइन किया है; आम तौर पर, यदि निवास सीओ के 5 से 10 मील के भीतर स्थित नहीं है, तो निवास को इंटरनेट के वैकल्पिक रूप का सहारा लेना चाहिए

पहुँच।

जबकि डीएसएल टेल्को के मौजूदा स्थानीय टेलीफोन बुनियादी ढांचे का उपयोग करता है, केबल इंटरनेट एक्सेस केबल टेलीविजन कंपनी के मौजूदा केबल टेलीविजन बुनियादी ढांचे का उपयोग करता है।

केबल इंटरनेट एक्सेस उसी कंपनी से जो केबल टेलीविजन प्रदान करती है। जैसा कि चित्र 1.6 में दर्शाया गया है, फाइबर ऑप्टिक्स केबल हेड एंड को पड़ोस-स्तर के जंक्शनों से जोड़ता है, जहाँ से पारंपरिक कोएक्सियल केबल का उपयोग व्यक्तिगत घरों और अपार्टमेंट तक पहुँचने के लिए किया जाता है। प्रत्येक पड़ोस जंक्शन

आम तौर पर 500 से 5,000 घरों को सपोर्ट करता है। क्योंकि इस सिस्टम में फाइबर और कोएक्सियल केबल दोनों का इस्तेमाल किया जाता है, इसलिए इसे अक्सर हाइब्रिड फाइबर कोएक्स (HFC) कहा जाता है।



चित्र 1.6 एक हाइब्रिड फाइबर-कोएक्सियल एक्सेस नेटवर्क

केबल इंटरनेट एक्सेस के लिए विशेष मोडेम की आवश्यकता होती है, जिसे केबल मोडेम कहा जाता है। DSL मोडेम की तरह, केबल मोडेम भी एक अच्छा विकल्प है।

मॉडेम आम तौर पर एक बाहरी डिवाइस है और ईथरनेट पोर्ट के माध्यम से होम पीसी से जुड़ता है। (हम

अध्याय 6 में ईथरनेट के बारे में विस्तार से चर्चा की गई है ।) केबल हेड एंड पर, केबल मॉडेम टर्मिनेशन सिस्टम (CMTS) DSL नेटवर्क के DSLAM के समान कार्य करता है - कई डाउनस्ट्रीम घरों में केबल मॉडेम से भेजे गए एनालॉग सिग्नल को वापस डिजिटल प्रारूप में बदलना। केबल मॉडेम HFC नेटवर्क को दो चैनलों में विभाजित करते हैं, एक डाउनस्ट्रीम और एक अपस्ट्रीम चैनल। DSL की तरह, एक्सेस भी है

आमतौर पर असममित, डाउनस्ट्रीम चैनल को आमतौर पर उच्च संचरण दर आवंटित की जाती है

अपस्ट्रीम चैनल। DOCSIS 2.0 मानक 42.8 एमबीपीएस तक की डाउनस्ट्रीम दरों और 30.7 एमबीपीएस तक की अपस्ट्रीम दरों को परिभाषित करता है।

DSL नेटवर्क के मामले में, अधिकतम प्राप्त करने योग्य दर हो सकती है

कम अनुबंधित डेटा दरों या मीडिया हानि के कारण इसका एहसास नहीं हो सकेगा।

केबल इंटरनेट एक्सेस की एक महत्वपूर्ण विशेषता यह है कि यह एक साझा प्रसारण माध्यम है।

विशेष रूप से, हेड एंड द्वारा भेजा गया प्रत्येक पैकेट प्रत्येक लिंक पर प्रत्येक घर और प्रत्येक नेटवर्क तक डाउनस्ट्रीम यात्रा करता है।

होम द्वारा भेजा गया पैकेट अपस्ट्रीम चैनल पर हेड एंड तक जाता है। इस कारण से, यदि कई उपयोगकर्ता एक साथ डाउनस्ट्रीम चैनल पर एक वीडियो फ़ाइल डाउनलोड कर रहे हैं, तो प्रत्येक उपयोगकर्ता को अपनी वीडियो फ़ाइल प्राप्त करने की वास्तविक दर कुल केबल डाउनस्ट्रीम दर से काफी कम होगी। दूसरी ओर, यदि केवल कुछ सक्रिय उपयोगकर्ता हैं और वे सभी वेब सर्फिंग कर रहे हैं, तो प्रत्येक उपयोगकर्ता वास्तव में पूर्ण केबल डाउनस्ट्रीम दर पर वेब पेज प्राप्त कर सकता है, क्योंकि उपयोगकर्ता शायद ही कभी अनुरोध करेंगे

एक ही समय में एक वेब पेज। क्योंकि अपस्ट्रीम चैनल भी साझा किया जाता है, एक वितरित

संचारणों को समन्वित करने और टकरावों से बचने के लिए मल्टीपल एक्सेस प्रोटोकॉल की आवश्यकता होती है। (हम इस पर चर्चा करेंगे

(अध्याय 6 में टकराव के मुद्दे पर विस्तार से चर्चा की गई है ।)

हालाँकि DSL और केबल नेटवर्क वर्तमान में संयुक्त राज्य अमेरिका में आवासीय ब्रॉडबैंड पहुँच के 85 प्रतिशत से अधिक का प्रतिनिधित्व करते हैं, एक उभरती हुई तकनीक जो और भी अधिक गति प्रदान करती है वह है फाइबर टू द होम (FTTH) [FTTH काउंसिल 2016]। जैसा कि नाम से पता चलता है, FTTH अवधारणा सरल है - CO से सीधे घर तक एक ऑप्टिकल फाइबर पथ प्रदान करना। आज कई देशों में - जिनमें UAE, दक्षिण कोरिया, हांगकांग, जापान, सिंगापुर, ताइवान, लिथुआनिया और स्वीडन शामिल हैं - अब घरेलू हैं

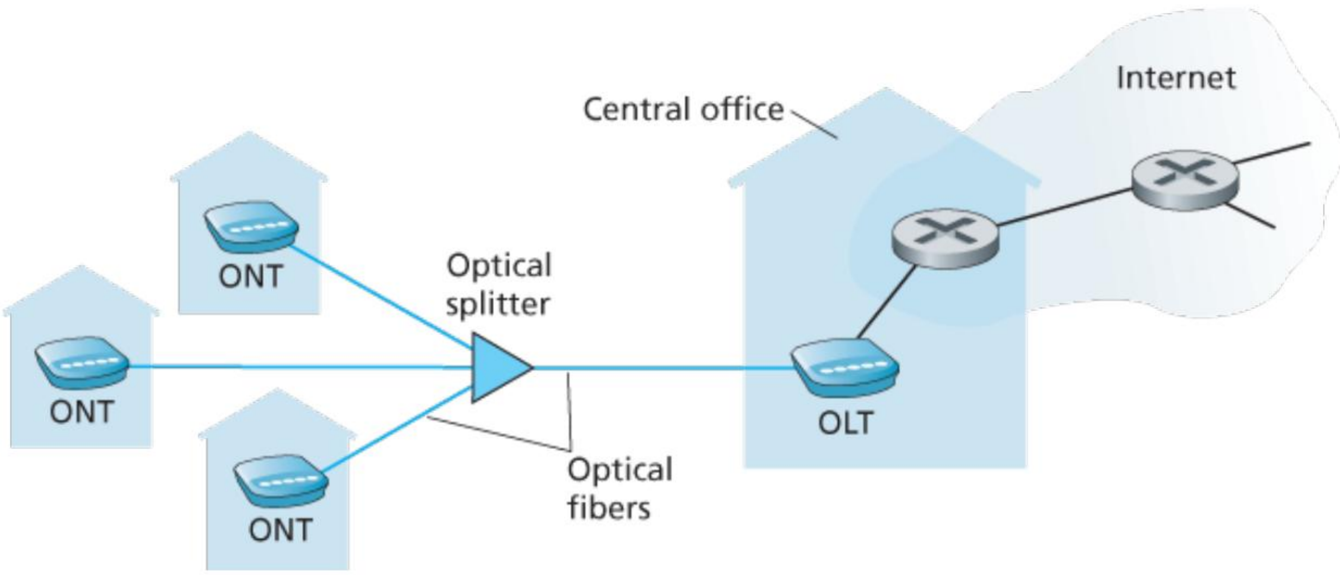
प्रवेश दर 30% से अधिक [एफटीटीएच परिषद 2016]।

CO से घरों तक ऑप्टिकल वितरण के लिए कई प्रतिस्पर्धी तकनीकें हैं। सबसे सरल ऑप्टिकल वितरण नेटवर्क को डायरेक्ट फाइबर कहा जाता है, जिसमें प्रत्येक घर के लिए CO से एक फाइबर निकलता है।

अधिक सामान्यतः, केंद्रीय कार्यालय से निकलने वाला प्रत्येक फाइबर वास्तव में कई घरों द्वारा साझा किया जाता है; यह तब तक नहीं होता है जब तक कि फाइबर घरों के अपेक्षाकृत करीब नहीं पहुंच जाता है कि इसे अलग-अलग ग्राहक-विशिष्ट फाइबर में विभाजित किया जाता है। दो प्रतिस्पर्धी ऑप्टिकल-वितरण नेटवर्क आर्किटेक्चर हैं जो इस विभाजन को निष्पादित करते हैं: सक्रिय ऑप्टिकल नेटवर्क (AON) और निष्क्रिय ऑप्टिकल नेटवर्क (PON)। AON अनिवार्य रूप से स्विच्ड ईथरनेट है, जो

अध्याय 6 में चर्चा की गई है ।

यहाँ, हम PON पर संक्षेप में चर्चा करते हैं, जिसका उपयोग Verizon की FIOS सेवा में किया जाता है। चित्र 1.7 PON वितरण आर्किटेक्चर का उपयोग करके FTTH को दर्शाता है। प्रत्येक घर में एक ऑप्टिकल नेटवर्क टर्मिनेटर (ONT) होता है, जो समर्पित ऑप्टिकल फाइबर द्वारा पड़ोस के स्प्लिटर से जुड़ा होता है। स्प्लिटर कई घरों (आमतौर पर कम) को जोड़ता है



चित्र 1.7 FTTH इंटरनेट एक्सेस

100 से अधिक) को एक एकल, साझा ऑप्टिकल फाइबर पर, जो टेल्को के सीओ में एक ऑप्टिकल लाइन टर्मिनेटर (ओएलटी) से जुड़ता है। ऑप्टिकल और इलेक्ट्रिकल सिग्नल के बीच रूपांतरण प्रदान करने वाला ओएलटी, टेल्को राउटर के माध्यम से इंटरनेट से जुड़ता है। घर में, उपयोगकर्ता एक होम राउटर (आमतौर पर एक वायरलेस राउटर) को ONT से जोड़ते हैं और इस होम राउटर के माध्यम से इंटरनेट का उपयोग करते हैं। PON आर्किटेक्चर में, OLT से स्प्लिटर को भेजे गए सभी पैकेट स्प्लिटर (केबल हेड एंड के समान) पर दोहराए जाते हैं।

FTTH संभावित रूप से गीगाबिट प्रति सेकंड रेंज में इंटरनेट एक्सेस दरें प्रदान कर सकता है। हालाँकि, अधिकांश FTTH ISP अलग-अलग दर की पेशकश करते हैं, जिसमें उच्च दर स्वाभाविक रूप से अधिक पैसे खर्च करती है। 2011 में यूएस FTTH ग्राहकों की औसत डाउनस्ट्रीम स्पीड लगभग 20 एमबीपीएस थी (केबल एक्सेस नेटवर्क के लिए 13 एमबीपीएस और डीएसएल के लिए 5 एमबीपीएस से कम की तुलना में) [FTTH काउंसिल 2011बी]।

घर तक इंटरनेट पहुँच प्रदान करने के लिए दो अन्य एक्सेस नेटवर्क तकनीकों का भी उपयोग किया जाता है। ऐसे स्थानों पर जहाँ DSL, केबल और FTTH उपलब्ध नहीं हैं (जैसे, कुछ ग्रामीण क्षेत्रों में), एक सैटेलाइट लिंक का उपयोग 1 एमबीपीएस से अधिक की गति से निवास को इंटरनेट से जोड़ने के लिए किया जा सकता है; स्टारबैंड और ह्यूजेसनेट दो ऐसे सैटेलाइट एक्सेस प्रदाता हैं। पारंपरिक फ़ोन लाइनों पर डायल-अप एक्सेस DSL के समान मॉडल पर आधारित है - एक होम मॉडेम एक फ़ोन लाइन के माध्यम से ISP में एक मॉडेम से जुड़ता है।

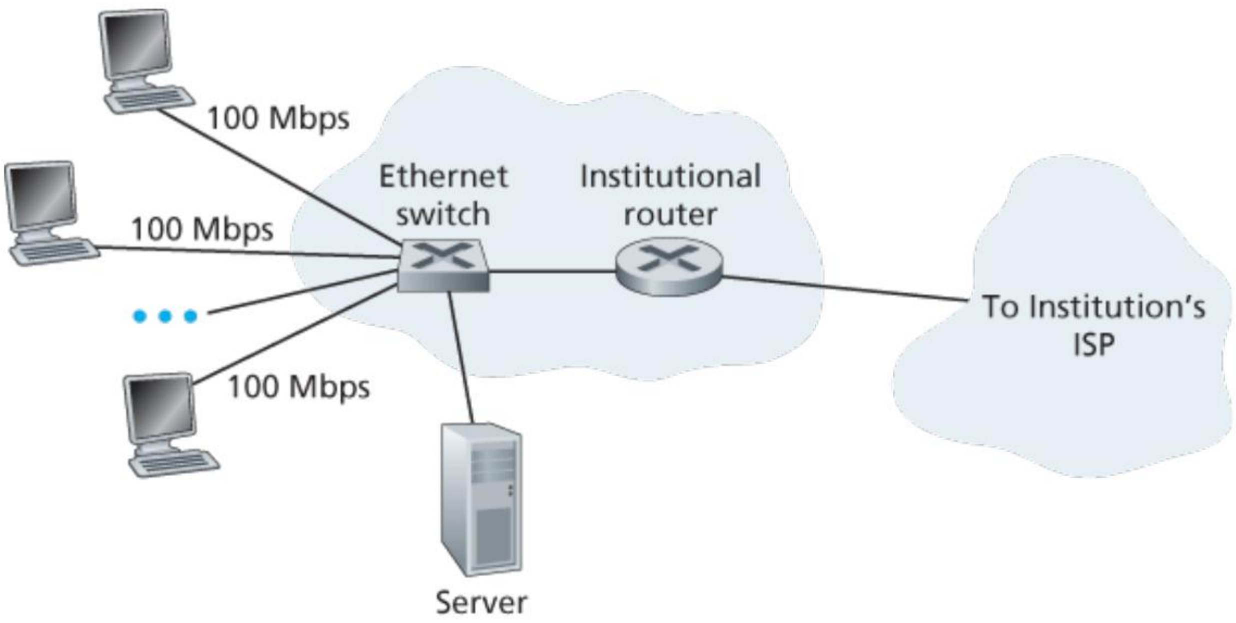
डीएसएल और अन्य ब्रॉडबैंड एक्सेस नेटवर्क की तुलना में, डायल-अप एक्सेस 56 केबीपीएस पर अत्यंत धीमी है।

उद्यम (और घर) में पहुंच: ईथरनेट और वाईफाई

कॉर्पोरेट और विश्वविद्यालय परिसरों में, और तेजी से घरेलू सेटिंग्स में, एक स्थानीय क्षेत्र नेटवर्क (LAN) का उपयोग अंतिम सिस्टम को एज राउटर से जोड़ने के लिए किया जाता है। हालाँकि LAN तकनीक के कई प्रकार हैं, लेकिन कॉर्पोरेट, विश्वविद्यालय और घरेलू नेटवर्क में ईथरनेट अब तक की सबसे प्रचलित एक्सेस तकनीक है।

चित्र 1.8 में दिखाए गए अनुसार , ईथरनेट उपयोगकर्ता ईथरनेट स्विच से कनेक्ट करने के लिए ट्विस्टेड-पेयर कॉपर वायर का उपयोग करते हैं, एक तकनीक जिस पर अध्याय 6 में विस्तार से चर्चा की गई है। ईथरनेट स्विच, या ऐसे नेटवर्क का उपयोग करने वाले

उपयोगकर्ता ईथरनेट स्विच से कनेक्ट करने के लिए ट्विस्टेड-पेयर कॉपर वायर का उपयोग करते हैं, एक तकनीक जिस पर **अध्याय 6 में विस्तार से चर्चा की गई है।**



चित्र 1.8 ईथरनेट इंटरनेट एक्सेस

इंटरकनेक्टेड स्विच, फिर बदले में बड़े इंटरनेट से जुड़ जाता है। ईथरनेट एक्सेस के साथ, उपयोगकर्ताओं के पास आमतौर पर ईथरनेट स्विच तक 100 एमबीपीएस या 1 जीबीपीएस की पहुंच होती है, जबकि सर्वर के पास 1 जीबीपीएस या 10 जीबीपीएस की पहुंच भी हो सकती है।

हालाँकि, तेजी से लोग लैपटॉप, स्मार्टफोन, टैबलेट से वायरलेस तरीके से इंटरनेट का उपयोग कर रहे हैं।

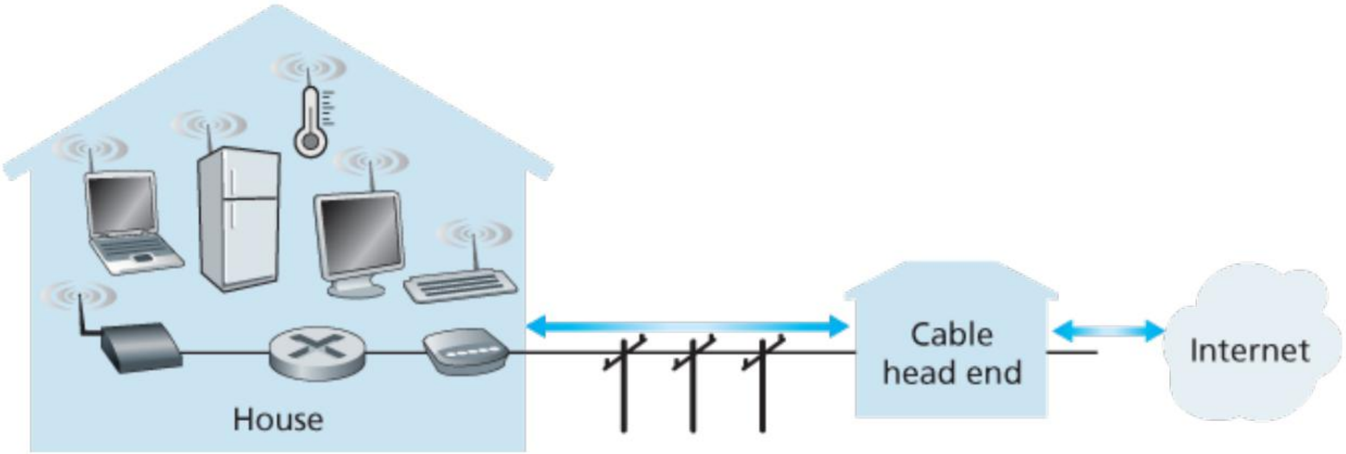
और अन्य "चीजें" (देखें "इंटरनेट ऑफ थिंग्स" पर पिछला साइडबार)। वायरलेस LAN सेटिंग में, वायरलेस उपयोगकर्ता एक एक्सेस पॉइंट से/से पैकेट संचारित/प्राप्त करते हैं जो एंटरप्राइज़ के नेटवर्क से जुड़ा होता है (सबसे अधिक संभावना वायर्ड ईथरनेट का उपयोग करके), जो बदले में वायर्ड इंटरनेट से जुड़ा होता है। एक वायरलेस LAN उपयोगकर्ता को आम तौर पर एक्सेस पॉइंट के कुछ दसियों मीटर के दायरे में होना चाहिए। IEEE 802.11 तकनीक पर आधारित वायरलेस LAN एक्सेस, जिसे आम बोलचाल की भाषा में WiFi के रूप में जाना जाता है, अब लगभग हर जगह है—विश्वविद्यालय, व्यावसायिक कार्यालय, कैफ़े, हवाई अड्डे, घर और यहाँ तक कि हवाई जहाज़ में भी। कई शहरों में, कोई सड़क के कोने पर खड़ा होकर दस या बीस बेस स्टेशनों की रेंज में हो सकता है (802.11 बेस स्टेशनों के ब्राउज़ करने योग्य वैश्विक मानचित्र के लिए जिन्हें उन लोगों द्वारा वेब साइट पर खोजा और लॉग किया गया है जो इसका बहुत आनंद लेते हैं

ऐसी चीजें करने के लिए, [wigle.net 2016] देखें)। जैसा कि अध्याय 7 में विस्तार से चर्चा की गई है , 802.11 आज 100 एमबीपीएस से अधिक की साझा संचरण दर प्रदान करता है।

हालाँकि ईथरनेट और वाईफ़ाई एक्सेस नेटवर्क को शुरू में एंटरप्राइज़ (कॉर्पोरेट, विश्वविद्यालय) सेटिंग्स में तैनात किया गया था, लेकिन हाल ही में वे घरेलू नेटवर्क के अपेक्षाकृत सामान्य घटक बन गए हैं। कई घर इन सस्ती सुविधाओं के साथ ब्रॉडबैंड आवासीय पहुँच (यानी, केबल मोडेम या DSL) को जोड़ते हैं

शक्तिशाली होम नेटवर्क बनाने के लिए वायरलेस LAN तकनीकें [एडवर्ड्स 2011]। चित्र 1.9 एक सामान्य होम नेटवर्क दिखाता है। इस होम नेटवर्क में एक रोमिंग लैपटॉप के साथ-साथ एक वायर्ड पीसी भी शामिल है; एक बेस स्टेशन (वायरलेस एक्सेस पॉइंट), जो वायरलेस पीसी और घर में अन्य वायरलेस डिवाइस के साथ संचार करता है; एक केबल मॉडेम, जो इंटरनेट तक ब्रॉडबैंड पहुँच प्रदान करता है; और एक राउटर, जो बेस स्टेशन और स्थिर पीसी को केबल मॉडेम से जोड़ता है। यह नेटवर्क घर के सदस्यों को इंटरनेट तक ब्रॉडबैंड पहुँच प्रदान करता है, जिसमें एक सदस्य रोमिंग से दूसरे सदस्य के साथ रहता है।

रसोईघर से लेकर पिछवाड़े और शयन कक्षों तक।



चित्र 1.9 एक सामान्य घरेलू नेटवर्क

वाइड-एरिया वायरलेस एक्सेस: 3G और LTE

तेजी से, iPhone और Android डिवाइस जैसे उपकरणों का उपयोग संदेश भेजने, सोशल नेटवर्क में फ़ोटो साझा करने, फ़िल्में देखने और चलते-फिरते संगीत स्ट्रीम करने के लिए किया जा रहा है। ये डिवाइस सेलुलर नेटवर्क प्रदाता द्वारा संचालित बेस स्टेशन के माध्यम से पैकेट भेजने/प्राप्त करने के लिए सेलुलर टेलीफोनी के लिए उपयोग किए जाने वाले समान वायरलेस इंफ्रास्ट्रक्चर का उपयोग करते हैं। WiFi के विपरीत, उपयोगकर्ता को बेस स्टेशन के कुछ दसियों किलोमीटर (कुछ दसियों मीटर के विपरीत) के भीतर ही रहने की आवश्यकता होती है।

दूरसंचार कंपनियों ने तथाकथित तीसरी पीढ़ी (3G) वायरलेस में भारी निवेश किया है, जो 1 एमबीपीएस से अधिक की गति पर पैकेट-स्विच्ड वाइड-एरिया वायरलेस इंटरनेट एक्सेस प्रदान करता है। लेकिन इससे भी अधिक गति वाली वाइड-एरिया एक्सेस तकनीकें - वाइड-एरिया वायरलेस नेटवर्क की चौथी पीढ़ी (4G) - पहले से ही तैनात की जा रही हैं। LTE (लॉन्ग-टर्म इवोल्यूशन के लिए - बैड एक्रोनिम ऑफ द ईयर अवार्ड के लिए एक उम्मीदवार) की जड़ें 3G तकनीक में हैं, और यह 10 एमबीपीएस से अधिक की दर प्राप्त कर सकता है। वाणिज्यिक तैनाती में कई दसियों एमबीपीएस की LTE डाउनस्ट्रीम दरें बताई गई हैं।

हम वायरलेस नेटवर्क और गतिशीलता के बुनियादी सिद्धांतों के साथ-साथ WiFi, 3G और LTE को भी कवर करेंगे

अध्याय 7 में प्रौद्योगिकियों (और अधिक!) के बारे में बताया गया है ।

1.2.2 भौतिक मीडिया

पिछले उपखंड में, हमने इंटरनेट में कुछ सबसे महत्वपूर्ण नेटवर्क एक्सेस तकनीकों का अवलोकन दिया था। जैसा कि हमने इन तकनीकों का वर्णन किया, हमने उपयोग किए जाने वाले भौतिक मीडिया का भी संकेत दिया। उदाहरण के लिए, हमने कहा कि HFC फाइबर केबल और कोएक्सियल केबल के संयोजन का उपयोग करता है। हमने कहा कि DSL और ईथरनेट तांबे के तार का उपयोग करते हैं। और हमने कहा कि मोबाइल एक्सेस नेटवर्क रेडियो स्पेक्ट्रम का उपयोग करते हैं। इस उपखंड में हम इन और अन्य ट्रांसमिशन मीडिया का संक्षिप्त अवलोकन प्रदान करते हैं जो आम तौर पर उपयोग किए जाते हैं

इंटरनेट में उपयोग किया जाता है.

भौतिक माध्यम से क्या अभिप्राय है, यह परिभाषित करने के लिए आइए बिट के संक्षिप्त जीवन पर विचार करें। लिंक और राउटर की एक श्रृंखला के माध्यम से एक छोर प्रणाली से दूसरे छोर प्रणाली तक यात्रा करने वाले बिट पर विचार करें। इस बेचारे बिट को कई बार इधर-उधर घुमाया जाता है और संचारित किया जाता है! स्रोत छोर प्रणाली पहले बिट संचारित करती है, और उसके तुरंत बाद श्रृंखला में पहला राउटर बिट प्राप्त करता है; फिर पहला राउटर बिट संचारित करता है, और उसके तुरंत बाद दूसरा राउटर बिट प्राप्त करता है; और इसी तरह। इस प्रकार हमारा बिट, स्रोत से गंतव्य तक यात्रा करते समय, ट्रांसमीटर-रिसीवर जोड़ों की एक श्रृंखला से गुजरता है। प्रत्येक ट्रांसमीटर-रिसीवर जोड़ी के लिए, बिट को भौतिक माध्यम में विद्युत चुम्बकीय तरंगों या ऑप्टिकल पल्स का प्रसार करके भेजा जाता है। भौतिक माध्यम कई आकार और रूप ले सकता है और पथ के साथ प्रत्येक ट्रांसमीटर-रिसीवर जोड़ी के लिए एक ही प्रकार का होना आवश्यक नहीं है। भौतिक मीडिया के उदाहरणों में मुड़-जोड़ी तांबे का तार, समाक्षीय केबल, मल्टीमोड फाइबर-ऑप्टिक केबल, स्थलीय रेडियो स्पेक्ट्रम और उपग्रह रेडियो स्पेक्ट्रम शामिल हैं। भौतिक मीडिया दो श्रेणियों में आते हैं: निर्देशित मीडिया और गैर निर्देशित मीडिया। निर्देशित मीडिया के साथ, तरंगों को एक ठोस माध्यम, जैसे कि फाइबर-ऑप्टिक केबल, एक मुड़-जोड़ी तांबे के तार, या एक समाक्षीय केबल के साथ निर्देशित किया जाता है। गैर निर्देशित मीडिया के साथ, तरंगें वायुमंडल और बाहरी अंतरिक्ष में फैलती हैं, जैसे कि वायरलेस लैन या डिजिटल सैटेलाइट चैनल में।

लेकिन इससे पहले कि हम विभिन्न मीडिया प्रकारों की विशेषताओं पर चर्चा करें, आइए हम उनकी लागतों के बारे में कुछ शब्द कहें। भौतिक लिंक (तांबे के तार, फाइबर-ऑप्टिक केबल, इत्यादि) की वास्तविक लागत अक्सर अन्य नेटवर्किंग लागतों की तुलना में अपेक्षाकृत कम होती है। विशेष रूप से, भौतिक लिंक की स्थापना से जुड़ी श्रम लागत सामग्री की लागत से कई गुना अधिक हो सकती है। इस कारण से, कई बिल्डर इमारत के हर कमरे में ट्विस्टेड पेयर, ऑप्टिकल फाइबर और कोएक्सियल केबल लगाते हैं। भले ही शुरुआत में केवल एक माध्यम का उपयोग किया जाता है, लेकिन इस बात की अच्छी संभावना है कि निकट भविष्य में किसी अन्य माध्यम का उपयोग किया जा सकता है, और इसलिए भविष्य में अतिरिक्त तार न बिछाने से पैसे की बचत होती है।

मुड़-जोड़ी तांबे का तार

सबसे कम खर्चीला और सबसे अधिक इस्तेमाल किया जाने वाला निर्देशित संचरण माध्यम ट्विस्टेड-पेयर कॉपर वायर है। सौ से ज़्यादा सालों से इसका इस्तेमाल टेलीफ़ोन नेटवर्क द्वारा किया जा रहा है। दरअसल, टेलीफ़ोन हैंडसेट से लेकर लोकल टेलीफ़ोन स्विच तक के 99 प्रतिशत से ज़्यादा वायर्ड कनेक्शन में ट्विस्टेड-पेयर कॉपर वायर का इस्तेमाल होता है। हममें से ज़्यादातर लोगों ने अपने घरों (या अपने माता-पिता या दादा-दादी के घरों!) और काम के माहौल में ट्विस्टेड पेयर देखा होगा। ट्विस्टेड पेयर में दो इंसुलेटेड कॉपर वायर होते हैं, जिनमें से प्रत्येक लगभग 1 मिमी मोटा होता है, जो एक नियमित सर्पिल पैटर्न में व्यवस्थित होता है। पास-पास के समान जोड़ों से विद्युत हस्तक्षेप को कम करने के लिए तारों को एक साथ घुमाया जाता है। आम तौर पर, कई जोड़ों को केबल में लपेटकर एक साथ बांधा जाता है

सुरक्षा कवच में जोड़े। एक तार जोड़ी एक एकल संचार लिंक का गठन करती है। अप्रतिरक्षित मुड़ जोड़ी (UTP) का उपयोग आम तौर पर किसी इमारत के अंदर कंप्यूटर नेटवर्क के लिए किया जाता है, यानी LAN के लिए। ट्विस्टेड पेयर का उपयोग करने वाले LAN के लिए डेटा दरें आज 10 एमबीपीएस से लेकर 10 जीबीपीएस तक होती हैं। जो डेटा दरें प्राप्त की जा सकती हैं, वे तार की मोटाई और ट्रांसमीटर और रिसीवर के बीच की दूरी पर निर्भर करती हैं।

जब 1980 के दशक में फाइबर-ऑप्टिक तकनीक उभरी, तो कई लोगों ने इसकी अपेक्षाकृत कम बिट दरों के कारण ट्विस्टेड पेयर की निंदा की। कुछ लोगों को तो यह भी लगा कि फाइबर-ऑप्टिक तकनीक पूरी तरह से बदल देगी
ट्विस्टेड पेयर। लेकिन ट्विस्टेड पेयर ने इतनी आसानी से हार नहीं मानी। आधुनिक ट्विस्टेड-पेयर तकनीक, जैसे कि श्रेणी

6a केबल, सौ मीटर तक की दूरी के लिए 10 Gbps की डेटा दर प्राप्त कर सकता है। अंत में, ट्विस्टेड पेयर हाई-स्पीड LAN नेटवर्किंग के लिए प्रमुख समाधान के रूप में उभरा है।

जैसा कि पहले चर्चा की गई है, ट्विस्टेड पेयर का उपयोग आम तौर पर आवासीय इंटरनेट एक्सेस के लिए भी किया जाता है। हमने देखा कि डायल-अप मॉडेम तकनीक ट्विस्टेड पेयर पर 56 केबीपीएस तक की दर से एक्सेस सक्षम करती है। हमने यह भी देखा कि डीएसएल (डिजिटल सब्सक्राइबर लाइन) तकनीक ने आवासीय उपयोगकर्ताओं को ट्विस्टेड पेयर पर दसियों एमबीपीएस पर इंटरनेट एक्सेस करने में सक्षम बनाया है (जब उपयोगकर्ता आईएसपी के केंद्रीय कार्यालय के करीब रहते हैं)।

समाक्षीय केबल

ट्विस्टेड पेयर की तरह, कोएक्सियल केबल में दो कॉपर कंडक्टर होते हैं, लेकिन दोनों कंडक्टर समानांतर होने के बजाय संकेंद्रित होते हैं। इस निर्माण और विशेष इन्सुलेशन और परिरक्षण के साथ, कोएक्सियल केबल उच्च डेटा संचरण दर प्राप्त कर सकता है। केबल टेलीविज़न सिस्टम में कोएक्सियल केबल काफी आम है। जैसा कि हमने पहले देखा, केबल टेलीविज़न सिस्टम को हाल ही में केबल मॉडेम के साथ जोड़ा गया है ताकि आवासीय उपयोगकर्ताओं को दसियों एमबीपीएस की दर से इंटरनेट एक्सेस प्रदान किया जा सके। केबल टेलीविज़न और केबल इंटरनेट एक्सेस में, ट्रांसमीटर डिजिटल सिग्नल को एक विशिष्ट आवृत्ति बैंड में स्थानांतरित करता है, और परिणामी एनालॉग सिग्नल ट्रांसमीटर से एक या अधिक रिसीवर को भेजा जाता है। कोएक्सियल केबल का उपयोग एक निर्देशित साझा माध्यम के रूप में किया जा सकता है। विशेष रूप से, कई अंत प्रणालियों को सीधे केबल से जोड़ा जा सकता है, जिसमें प्रत्येक अंत प्रणाली दूसरे अंत प्रणालियों द्वारा भेजी गई हर चीज़ को प्राप्त करती है।

फाइबर ऑप्टिक्स

ऑप्टिकल फाइबर एक पतला, लचीला माध्यम है जो प्रकाश की तरंगों का संचालन करता है, जिसमें प्रत्येक तरंग एक बिट का प्रतिनिधित्व करती है। एक एकल ऑप्टिकल फाइबर प्रति सेकंड दसियों या सैकड़ों गीगाबिट तक की जबरदस्त बिट दरों का समर्थन कर सकता है। वे विद्युत चुम्बकीय हस्तक्षेप से प्रतिरक्षित हैं, 100 किलोमीटर तक बहुत कम सिग्नल क्षीणन है, और उन्हें टैप करना बहुत कठिन है। इन विशेषताओं ने फाइबर ऑप्टिक्स को विशेष रूप से विदेशी लिंक के लिए पसंदीदा लंबी दूरी का निर्देशित ट्रांसमिशन मीडिया बना दिया है। संयुक्त राज्य अमेरिका और अन्य जगहों पर कई लंबी दूरी के टेलीफोन नेटवर्क अब विशेष रूप से फाइबर ऑप्टिक्स का उपयोग करते हैं। फाइबर ऑप्टिक्स इंटरनेट की रीढ़ में भी प्रचलित है। हालाँकि, ऑप्टिकल डिवाइस की उच्च लागत - जैसे ट्रांसमीटर, रिसीवर और स्विच - ने शॉर्ट-हॉल ट्रांसपोर्ट के लिए उनकी तैनाती में बाधा डाली है, जैसे कि LAN में या आवासीय एक्सेस नेटवर्क में घर में। ऑप्टिकल कैरियर (OC) मानक लिंक गति

51.8 एमबीपीएस से लेकर 39.8 जीबीपीएस तक; इन विनिर्देशों को अक्सर OC-n के रूप में संदर्भित किया जाता है, जहाँ लिंक स्पीड n □ 51.8 एमबीपीएस के बराबर होती है। आज उपयोग में आने वाले मानकों में OC-1, OC-3, OC-12, OC-24, OC-48, OC-96, OC-192, OC-768 शामिल हैं। [मुखर्जी 2006, रामास्वामी 2010] ऑप्टिकल नेटवर्किंग के विभिन्न पहलुओं की कवरेज प्रदान करते हैं।

स्थलीय रेडियो चैनल

रेडियो चैनल विद्युत चुम्बकीय स्पेक्ट्रम में सिग्नल ले जाते हैं। वे एक आकर्षक माध्यम हैं क्योंकि उन्हें स्थापित करने के लिए किसी भौतिक तार की आवश्यकता नहीं होती है, वे दीवारों में प्रवेश कर सकते हैं, मोबाइल उपयोगकर्ता को कनेक्टिविटी प्रदान कर सकते हैं,

और संभावित रूप से लंबी दूरी तक सिग्नल ले जा सकता है। रेडियो चैनल की विशेषताएं प्रसार वातावरण और सिग्नल को ले जाने वाली दूरी पर काफी हद तक निर्भर करती हैं।

पर्यावरणीय विचार पथ हानि और छाया क्षीणता (जो सिग्नल की शक्ति को कम कर देता है, क्योंकि सिग्नल एक दूरी तक और अवरोधक वस्तुओं के चारों ओर/से होकर यात्रा करता है), बहुपथ क्षीणता (हस्तक्षेप करने वाली वस्तुओं से सिग्नल के परावर्तन के कारण) और हस्तक्षेप (अन्य प्रसारणों और विद्युत चुम्बकीय संकेतों के कारण) निर्धारित करते हैं।

स्थलीय रेडियो चैनलों को मोटे तौर पर तीन समूहों में वर्गीकृत किया जा सकता है: वे जो बहुत कम दूरी पर संचालित होते हैं (जैसे, एक या दो मीटर के साथ); वे जो स्थानीय क्षेत्रों में संचालित होते हैं, जो आम तौर पर दस से कुछ सौ मीटर तक फैले होते हैं; और वे जो विस्तृत क्षेत्र में संचालित होते हैं, जो दसियों किलोमीटर तक फैले होते हैं। वायरलेस हेडसेट, कीबोर्ड और मेडिकल डिवाइस जैसे व्यक्तिगत उपकरण कम दूरी पर संचालित होते हैं;

धारा 1.2.1 में वर्णित वायरलेस LAN प्रौद्योगिकियाँ स्थानीय-क्षेत्र रेडियो चैनलों का उपयोग करती हैं; सेलुलर एक्सेस प्रौद्योगिकियाँ व्यापक-क्षेत्र रेडियो चैनलों का उपयोग करती हैं। हम अध्याय 7 में रेडियो चैनलों पर विस्तार से चर्चा करेंगे।

सैटेलाइट रेडियो चैनल

संचार उपग्रह दो या अधिक पृथ्वी-आधारित माइक्रोवेव ट्रांसमीटर/रिसीवर को जोड़ता है, जिन्हें ग्राउंड स्टेशन के रूप में जाना जाता है। उपग्रह एक आवृत्ति बैंड पर प्रसारण प्राप्त करता है, एक रिपीटर (नीचे चर्चा की गई) का उपयोग करके सिग्नल को पुन: उत्पन्न करता है, और दूसरी आवृत्ति पर सिग्नल संचारित करता है। संचार में दो प्रकार के उपग्रहों का उपयोग किया जाता है: भूस्थिर उपग्रह और निम्न-पृथ्वी परिक्रमा (LEO)

उपग्रह [विकी सैटेलाइट 2016].

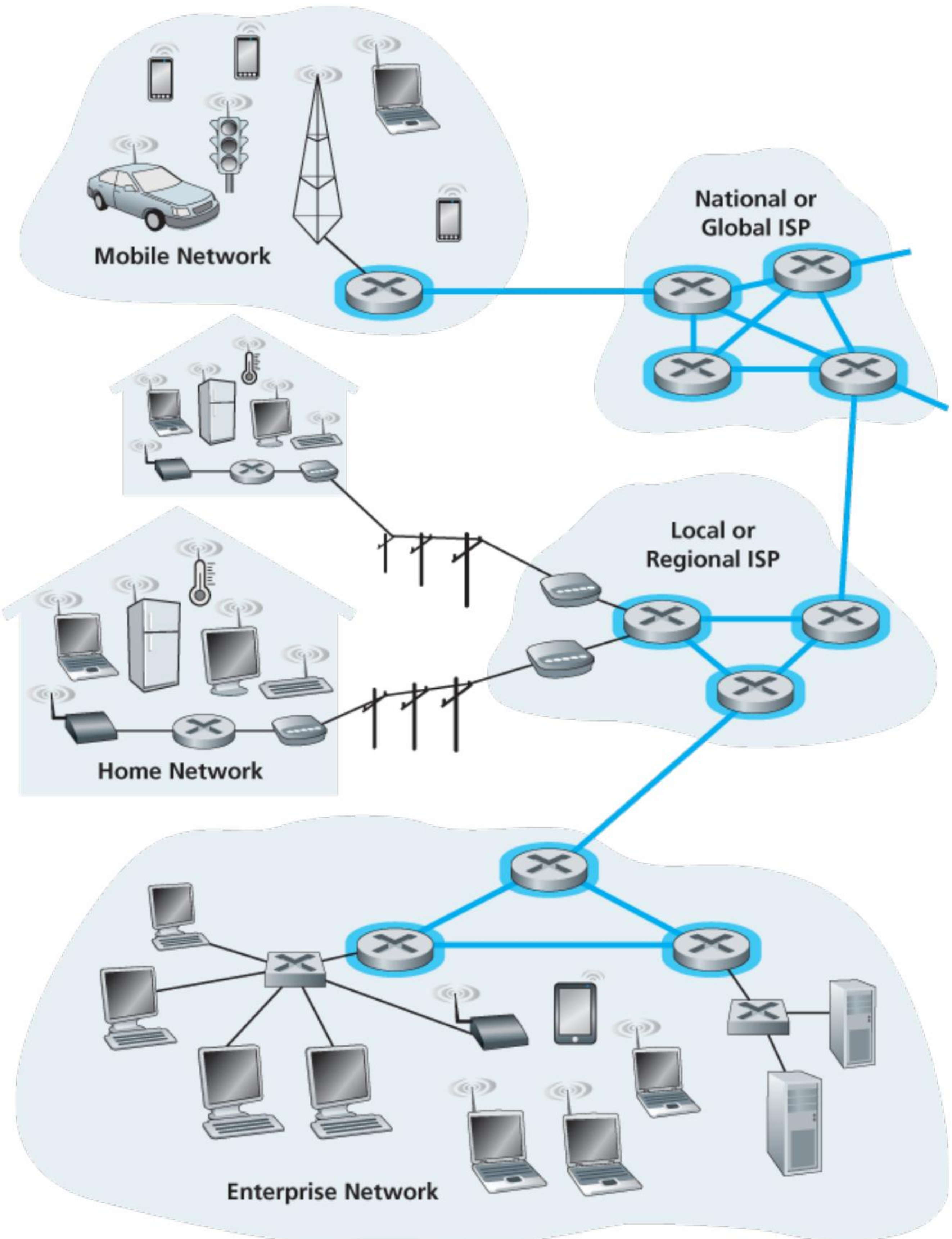
भूस्थिर उपग्रह पृथ्वी पर एक ही स्थान पर स्थायी रूप से बने रहते हैं। यह स्थिर उपस्थिति उपग्रह को पृथ्वी की सतह से 36,000 किलोमीटर ऊपर कक्षा में स्थापित करके प्राप्त की जाती है। ग्राउंड स्टेशन से सैटेलाइट के माध्यम से ग्राउंड स्टेशन तक की यह बड़ी दूरी 280 मिलीसेकंड की पर्याप्त सिग्नल प्रसार देरी का कारण बनती है। फिर भी, सैटेलाइट लिंक, जो सैकड़ों एमबीपीएस की गति से काम कर सकते हैं, अक्सर उन क्षेत्रों में उपयोग किए जाते हैं जहां डीएसएल या केबल-आधारित इंटरनेट एक्सेस तक पहुंच नहीं है।

LEO उपग्रहों को पृथ्वी के बहुत करीब रखा जाता है तथा वे पृथ्वी पर एक स्थान के ऊपर स्थायी रूप से नहीं रहते हैं।
वे पृथ्वी के चारों ओर घूमते हैं (जैसे चंद्रमा घूमता है) और एक दूसरे के साथ संवाद भी कर सकते हैं।
ग्राउंड स्टेशनों के साथ। किसी क्षेत्र को निरंतर कवरेज प्रदान करने के लिए, कई उपग्रहों को जमीन पर स्थापित करने की आवश्यकता होती है
कक्षा में। वर्तमान में कई निम्न-ऊंचाई संचार प्रणालियाँ विकास के चरण में हैं। भविष्य में कभी-कभी इंटरनेट एक्सेस के लिए LEO उपग्रह प्रौद्योगिकी का उपयोग किया जा सकता है।

1.3 नेटवर्क कोर

इंटरनेट के किनारे की जांच करने के बाद, आइए अब नेटवर्क कोर - जाल के अंदर और अधिक गहराई से जाएं

पैकेट स्विच और लिंक का वह समूह जो इंटरनेट के अंतिम सिस्टम को आपस में जोड़ता है। चित्र 1.10 नेटवर्क कोर को मोटी, छायांकित रेखाओं से हाइलाइट करता है।



चित्र 1.10 नेटवर्क कोर

1.3.1 पैकेट स्विचिंग

नेटवर्क एप्लीकेशन में, एंड सिस्टम **एक दूसरे के साथ** संदेशों का **आदान-प्रदान करते हैं** । संदेशों में वह सब कुछ हो सकता है जो एप्लीकेशन डिजाइनर चाहता है। संदेश एक नियंत्रण फ़ंक्शन कर सकते हैं (उदाहरण के लिए, "हाय"

चित्र 1.2 में हमारे हैंडशेकिंग उदाहरण में संदेश) या इसमें डेटा हो सकता है, जैसे कि एक ई-मेल संदेश, एक JPEG छवि, या एक MP3 ऑडियो फ़ाइल। स्रोत अंत प्रणाली से गंतव्य अंत प्रणाली तक संदेश भेजने के लिए, स्रोत लंबे संदेशों को डेटा के छोटे टुकड़ों में तोड़ देता है जिन्हें पैकेट के रूप में जाना जाता है। स्रोत और गंतव्य के बीच, प्रत्येक पैकेट संचार लिंक और पैकेट स्विच (जिसके लिए दो प्रमुख प्रकार हैं, राउटर और लिंक-लेयर स्विच) के माध्यम से यात्रा करता है। पैकेट को संचारित किया जाता है

प्रत्येक संचार लिंक को लिंक की पूर्ण **संचरण दर** के बराबर दर पर भेजा जाता है **। इसलिए, यदि कोई स्रोत अंत प्रणाली या पैकेट स्विच, संचरण दर** R **बिट्स/सेकंड** वाले लिंक पर L **बिट्स का पैकेट भेज रहा है ,** तो पैकेट को संचारित करने में लगने वाला समय L/R सेकंड है।

स्टोर-एंड-फॉरवर्ड ट्रांसमिशन

अधिकांश पैकेट स्विच **लिंक के इनपुट पर** स्टोर-एंड-फॉरवर्ड ट्रांसमिशन **का उपयोग करते हैं** । स्टोर-एंड-फॉरवर्ड ट्रांसमिशन का मतलब है कि पैकेट स्विच को आउटबाउंड लिंक पर पैकेट के पहले बिट को संचारित करने से पहले पूरे पैकेट को प्राप्त करना होगा। स्टोर-एंड-फॉरवर्ड ट्रांसमिशन को अधिक विस्तार से जानने के लिए, एक सरल नेटवर्क पर विचार करें जिसमें एक राउटर द्वारा जुड़े दो एंड सिस्टम शामिल हैं, जैसा कि दिखाया गया है

चित्र 1.11 में । राउटर में आमतौर पर कई इंसिडेंट लिंक होते हैं, क्योंकि इसका काम इनकमिंग पैकेट को आउटगोइंग लिंक पर स्विच करना होता है; इस सरल उदाहरण में, राउटर के पास एक पैकेट को एक (इनपुट) लिंक से दूसरे अटैच लिंक पर ट्रांसफर करने का सरल कार्य होता है। इस उदाहरण में, स्रोत में तीन हैं

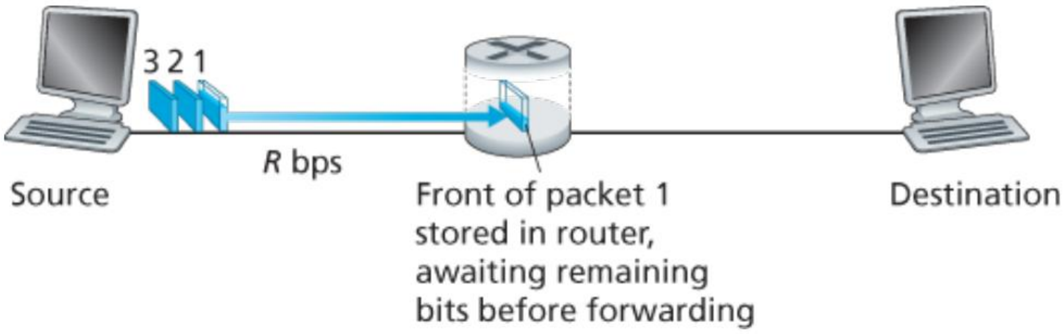
पैकेट, जिनमें से प्रत्येक में L **बिट्स होते हैं, गंतव्य पर भेजने के लिए।** चित्र 1.11 में दिखाए गए समय के स्नैपशॉट पर , स्रोत ने पैकेट 1 का कुछ हिस्सा प्रेषित कर दिया है, और पैकेट 1 का अगला हिस्सा पहले ही राउटर पर पहुँच चुका है। चूँकि राउटर स्टोर-एंड-फ़ॉरवर्डिंग का उपयोग करता है, इसलिए इस समय, राउटर अपने द्वारा प्राप्त बिट्स को प्रेषित नहीं कर सकता है; इसके बजाय उसे पहले पैकेट के बिट्स को बफर (यानी, "स्टोर") करना होगा। केवल उसके बाद

राउटर को **पैकेट के** सभी बिट्स प्राप्त हो जाने के बाद, वह पैकेट को आउटबाउंड लिंक पर भेजना (यानी, "फॉरवर्ड") शुरू कर सकता है। स्टोर-एंड-फॉरवर्ड ट्रांसमिशन के बारे में कुछ जानकारी प्राप्त करने के लिए, आइए अब उस समय की मात्रा की गणना करें जो स्रोत द्वारा पैकेट भेजना शुरू करने से लेकर गंतव्य तक पूरा पैकेट प्राप्त करने तक बीतता है। (यहां हम प्रसार विलंब को अनदेखा करेंगे - बिट्स को यात्रा करने में लगने वाला समय

प्रकाश की गति के निकट तार पर - जिस पर अनुभाग 1.4 में चर्चा की जाएगी।) स्रोत समय 0 पर संचार करना शुरू करता है; समय L/R सेकंड पर, स्रोत ने संपूर्ण पैकेट संचारित कर दिया है, और संपूर्ण पैकेट राउटर पर प्राप्त और संग्रहीत हो गया है (क्योंकि कोई प्रसार विलंब नहीं है)। समय L/R सेकंड पर, चूंकि राउटर ने अभी-अभी संपूर्ण पैकेट प्राप्त किया है, इसलिए यह पैकेट को राउटर पर संचारित करना शुरू कर सकता है।

आउटबाउंड लिंक को गंतव्य की ओर भेजना; समय 2L/R पर, राउटर ने संपूर्ण पैकेट प्रेषित कर दिया है, और

पूरा पैकेट गंतव्य द्वारा प्राप्त कर लिया गया है। इस प्रकार, कुल विलंब 2L/R है। यदि



चित्र 1.11 स्टोर-एंड-फॉरवर्ड पैकेट स्विचिंग

स्विच को आगे भेजे गए बिट्स को तुरंत ही स्विच करना चाहिए जैसे ही वे आते हैं (पूरे पैकेट को प्राप्त किए बिना), फिर

कुल विलंब L/R होगा क्योंकि बिट्स राउटर पर रुके नहीं रहते। लेकिन, जैसा कि हम अनुभाग 1.4 में चर्चा करेंगे, **राउटर** को अग्रेषित करने से पहले पूरे पैकेट को प्राप्त करने, संग्रहीत करने और संसाधित करने की **आवश्यकता होती है।**

अब आइए उस समय की गणना करें जो स्रोत द्वारा पहला पैकेट भेजना शुरू करने से लेकर बीतने तक बीतता है

जब तक कि गंतव्य को तीनों पैकेट प्राप्त न हो जाएं। पहले की तरह, L/R समय पर, राउटर पहले पैकेट को अग्रेषित करना शुरू कर देता है। लेकिन L/R **समय पर** स्रोत दूसरा पैकेट भेजना भी शुरू कर देगा, क्योंकि उसने अभी-अभी पूरा पहला पैकेट भेजना समाप्त किया है। इस प्रकार, 2L/R समय पर, गंतव्य को पहला पैकेट प्राप्त हो गया है और राउटर को दूसरा पैकेट प्राप्त हो गया है। इसी तरह, 3L/R समय पर, गंतव्य को पहले दो पैकेट प्राप्त हो गए हैं और राउटर को तीसरा पैकेट प्राप्त हो गया है। अंत में, 4L/R **समय पर** गंतव्य को तीनों पैकेट प्राप्त हो गए हैं!

आइए अब एक पैकेट को स्रोत से गंतव्य तक पथ पर भेजने के सामान्य मामले पर विचार करें

इसमें N लिंक होते हैं, जिनमें से प्रत्येक की दर R होती है (इस प्रकार, स्रोत और गंतव्य के बीच N-1 राउटर होते हैं)।
उपरोक्त तर्क को लागू करने पर, हम देखते हैं कि अंत-से-अंत तक विलंब है:

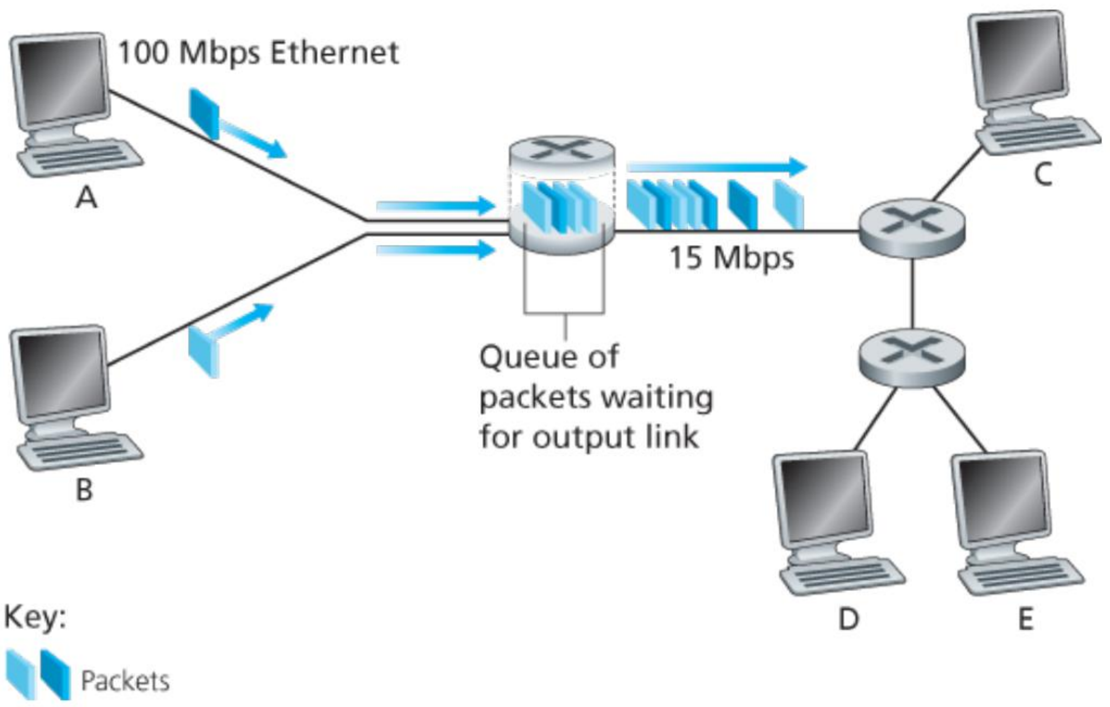
डेन्ड-टू-एंड=एनएलआर (1.1)

अब आप यह निर्धारित करने का प्रयास करना चाहेंगे कि N लिंकों की श्रृंखला पर भेजे गए P **पैकेटों** के लिए विलंब कितना होगा ।

कतार में देरी और पैकेट हानि

प्रत्येक पैकेट स्विच में कई लिंक जुड़े होते हैं। प्रत्येक जुड़े लिंक के लिए, पैकेट स्विच में एक आउटपुट बफर (जिसे आउटपुट कतार भी कहा जाता है) होता है, जो उन पैकेट को संग्रहीत करता है जिन्हें राउटर उस लिंक में भेजने वाला होता है। आउटपुट बफ़र्स पैकेट स्विचिंग में महत्वपूर्ण भूमिका निभाते हैं। यदि किसी आने वाले पैकेट को किसी लिंक पर प्रसारित करने की आवश्यकता है, लेकिन लिंक को किसी अन्य पैकेट के प्रसारण में व्यस्त पाता है, तो आने वाले पैकेट को आउटपुट बफर में प्रतीक्षा करनी चाहिए। इस प्रकार, स्टोर-एंड-फॉरवर्ड देरी के अलावा, पैकेट आउटपुट बफर कतार में देरी से पीड़ित होते हैं। ये देरी परिवर्तनशील हैं और नेटवर्क में भीड़ के स्तर पर निर्भर करती हैं।

चूंकि बफर स्पेस की मात्रा सीमित है,



चित्र 1.12 पैकेट स्विचिंग

आने वाले पैकेट को लग सकता है कि बफर पूरी तरह से अन्य पैकेटों से भरा हुआ है जो ट्रांसमिशन के लिए प्रतीक्षा कर रहे हैं। इस स्थिति में, पैकेट का नुकसान होगा - या तो आने वाला पैकेट या पहले से कतारबद्ध पैकेट में से एक गिरा दिया जाएगा।

चित्र 1.12 एक सरल पैकेट-स्विच्ड नेटवर्क को दर्शाता है। चित्र 1.11 की तरह, पैकेट को तीन-आयामी स्लैब द्वारा दर्शाया जाता है। स्लैब की चौड़ाई पैकेट में बिट्स की संख्या को दर्शाती है। इस चित्र में, सभी पैकेट की चौड़ाई समान है और इसलिए उनकी लंबाई भी समान है। मान लीजिए होस्ट A और B होस्ट E को पैकेट भेज रहे हैं। होस्ट A और B पहले अपने पैकेट को 100 एमबीपीएस ईथरनेट लिंक के साथ पहले राउटर पर भेजते हैं। राउटर फिर इन पैकेट को 15 एमबीपीएस लिंक पर निर्देशित करता है। यदि, थोड़े समय के अंतराल के दौरान, राउटर पर पैकेट की आगमन दर (जब प्रति सेकंड बिट्स में परिवर्तित की जाती है) 15 एमबीपीएस से अधिक हो जाती है, तो राउटर पर भीड़भाड़ हो जाएगी क्योंकि लिंक पर संचारित होने से पहले पैकेट लिंक के आउटपुट बफर में कतार में खड़े हो जाते हैं।

उदाहरण के लिए, यदि होस्ट A और B प्रत्येक एक ही समय में लगातार पाँच पैकेट भेजते हैं, तो इनमें से अधिकांश पैकेट कतार में प्रतीक्षा करने में कुछ समय बिताएँगे। स्थिति, वास्तव में, कई आम दिनों की स्थितियों के बिल्कुल समान है - उदाहरण के लिए, जब हम बैंक टेलर के लिए लाइन में प्रतीक्षा करते हैं या प्रतीक्षा करते हैं

टोल बूथ के सामने। हम इस कतार में देरी की अधिक विस्तार से अनुभाग 1.4 में जांच करेंगे।

अग्रेषण तालिकाएँ और रूटिंग प्रोटोकॉल

पहले हमने बताया कि राउटर अपने किसी संलग्न संचार लिंक पर आने वाले पैकेट को लेता है और उस पैकेट को अपने किसी संलग्न संचार लिंक पर अग्रेषित करता है। लेकिन राउटर यह कैसे निर्धारित करता है कि उसे पैकेट को किस लिंक पर अग्रेषित करना चाहिए? पैकेट अग्रेषण वास्तव में विभिन्न प्रकार के कंप्यूटर नेटवर्क में अलग-अलग तरीकों से किया जाता है। यहाँ, हम संक्षेप में वर्णन करते हैं कि यह इंटरनेट पर कैसे किया जाता है।

इंटरनेट में, हर एंड सिस्टम का एक पता होता है जिसे IP पता कहते हैं। जब कोई सोर्स एंड सिस्टम किसी पैकेट को गंतव्य एंड सिस्टम पर भेजना चाहता है, तो स्रोत पैकेट के हेडर में गंतव्य का IP पता शामिल करता है। डाक पते की तरह, इस पते में एक पदानुक्रमित संरचना होती है। जब कोई पैकेट नेटवर्क में राउटर पर पहुंचता है, तो राउटर पैकेट के गंतव्य पते के एक हिस्से की जांच करता है और पैकेट को बगल के राउटर को अग्रेषित करता है। अधिक विशेष रूप से, प्रत्येक राउटर में एक अग्रेषण तालिका **होती है** जो गंतव्य पते (या गंतव्य पते के कुछ हिस्सों) को उस राउटर के आउटबाउंड लिंक से मैप करती है।

जब कोई पैकेट राउटर पर पहुंचता है, तो राउटर उस पते की जांच करता है और उचित आउटबाउंड लिंक खोजने के लिए इस गंतव्य पते का उपयोग करके अपनी फ़ॉरवर्डिंग टेबल खोजता है। राउटर फिर पैकेट को निर्देशित करता है
इस आउटबाउंड लिंक पर जाएँ।

एंड-टू-एंड रूटिंग प्रक्रिया एक कार चालक के समान है जो नक्शे का उपयोग नहीं करता है, बल्कि दिशा-निर्देश पूछना पसंद करता है। उदाहरण के लिए, मान लीजिए कि जो फिलाडेल्फिया से ऑरलैंडो, फ्लोरिडा में 156 लेकसाइड ड्राइव तक गाड़ी चला रहा है। जो पहले अपने पड़ोस के गैस स्टेशन पर जाता है और पूछता है कि ऑरलैंडो, फ्लोरिडा में 156 लेकसाइड ड्राइव तक कैसे पहुंचा जाए। गैस स्टेशन अटेंडेंट पते के फ्लोरिडा वाले हिस्से को निकालता है और जो को बताता है कि उसे अंतरराज्यीय राजमार्ग I-95 साउथ पर जाना होगा, जिसका प्रवेश द्वार गैस स्टेशन के ठीक बगल में है। वह जो को यह भी बताता है कि एक बार जब वह फ्लोरिडा में प्रवेश कर जाता है, तो उसे वहां किसी और से पूछना चाहिए। फिर जो जैक्सनविले, फ्लोरिडा पहुंचने तक I-95 साउथ लेता है, जिस बिंदु पर वह दिशा-निर्देश के लिए दूसरे गैस स्टेशन अटेंडेंट से पूछता है। अटेंडेंट पते के ऑरलैंडो वाले हिस्से को निकालता है और जो को बताता है कि उसे डेटोना बीच तक I-95 पर चलते रहना चाहिए और फिर किसी और से पूछना चाहिए। डेटोना बीच में, दूसरा गैस स्टेशन अटेंडेंट भी पते के ऑरलैंडो वाले हिस्से को निकालता है और जो को बताता है कि उसे ऑरलैंडो के लिए सीधे I-4 लेना चाहिए। जो I-4 लेता है और ऑरलैंडो निकास पर उतर जाता है। जो दूसरे गैस स्टेशन अटेंडेंट के पास जाता है, और इस बार अटेंडेंट पते के लेकसाइड ड्राइव वाले हिस्से को निकालता है और जो को बताता है कि उसे लेकसाइड ड्राइव तक पहुँचने के लिए किस सड़क पर चलना चाहिए। जब जो लेकसाइड ड्राइव पर पहुँचता है, तो वह साइकिल पर सवार एक बच्चे से पूछता है कि वह अपने गंतव्य तक कैसे पहुँच सकता है। बच्चा पते के 156वें हिस्से को निकालता है और घर की ओर इशारा करता है। जो आखिरकार अपने अंतिम गंतव्य पर पहुँच जाता है। उपरोक्त सादृश्य में, गैस स्टेशन अटेंडेंट और साइकिल पर सवार बच्चे राउटर के समान हैं।

हमने अभी सीखा कि राउटर पैकेट के गंतव्य पते का उपयोग फ़ॉरवर्डिंग टेबल को इंडेक्स करने और उचित आउटबाउंड लिंक निर्धारित करने के लिए करता है। लेकिन यह कथन एक और सवाल उठाता है: फ़ॉरवर्डिंग टेबल कैसे सेट की जाती हैं? क्या वे प्रत्येक राउटर में हाथ से कॉन्फ़िगर की जाती हैं, या इंटरनेट

अधिक स्वचालित प्रक्रिया का उपयोग करें? इस मुद्दे का अध्याय 5 में गहराई से अध्ययन किया जाएगा। लेकिन यहाँ आपकी रुचि बढ़ाने के लिए, हम अब ध्यान देंगे कि इंटरनेट में कई विशेष रूटिंग प्रोटोकॉल **हैं** जिनका उपयोग फ़ॉरवर्डिंग टेबल को स्वचालित रूप से सेट करने के लिए किया जाता है। एक रूटिंग प्रोटोकॉल, उदाहरण के लिए, प्रत्येक राउटर से प्रत्येक गंतव्य तक सबसे छोटा रास्ता निर्धारित कर सकता है और फ़ॉरवर्डिंग को कॉन्फ़िगर करने के लिए सबसे छोटे पथ परिणामों का उपयोग कर सकता है

राउटर में तालिकाएँ.

आप वास्तव में यह देखना चाहेंगे कि इंटरनेट पर पैकेट किस तरह से एंड-टू-एंड रूट लेते हैं? अब हम आपको ट्रेस-रूट प्रोग्राम के साथ बातचीत करके अपने हाथों को गंदा करने के लिए आमंत्रित करते हैं। बस साइट पर जाएँ

www.traceroute.org, किसी विशेष देश में एक स्रोत चुनें, और उस स्रोत से अपने कंप्यूटर तक मार्ग का पता लगाएँ। (ट्रेसराउट पर चर्चा के लिए, अनुभाग 1.4 देखें।)

1.3.2 सर्किट स्विचिंग

लिंक और स्विच के नेटवर्क के माध्यम से डेटा को स्थानांतरित करने के दो मूलभूत तरीके हैं: सर्किट स्विचिंग और पैकेट स्विचिंग। पिछले लेख में पैकेट-स्विच्ड नेटवर्क को कवर किया गया है

उपधारा में, अब हम अपना ध्यान सर्किट-स्विच्ड नेटवर्क की ओर मोड़ते हैं।

सर्किट-स्विच्ड नेटवर्क में, पथ (बफर, लिंक ट्रांसमिशन दर) के साथ आवश्यक संसाधन

अंत प्रणालियों के बीच संचार के लिए प्रदान किए जाने वाले संसाधन अंत प्रणालियों के बीच संचार सत्र की अवधि के लिए आरक्षित **होते हैं। पैकेट-स्विच्ड नेटवर्क में, ये संसाधन आरक्षित** नहीं होते **हैं** ; एक सत्र के संदेश मांग पर संसाधनों का उपयोग करते हैं और, परिणामस्वरूप, संचार लिंक तक पहुंच के लिए प्रतीक्षा (यानी कतार) करनी पड़ सकती है। एक सरल सादृश्य के रूप में, दो रेस्तरां पर विचार करें, एक जिसे आरक्षण की आवश्यकता होती है और दूसरा जिसे न तो आरक्षण की आवश्यकता होती है और न ही उन्हें स्वीकार करता है। जिस रेस्तरां को आरक्षण की आवश्यकता होती है, उसके लिए हमें घर से निकलने से पहले कॉल करने की परेशानी से गुजरना पड़ता है।

लेकिन जब हम रेस्तरां में पहुंचते हैं तो सिद्धांततः हम तुरंत बैठ सकते हैं और अपने भोजन का ऑर्डर दे सकते हैं।

जिस रेस्टोरेंट में आरक्षण की आवश्यकता नहीं होती, वहां हमें टेबल आरक्षित करने की चिंता करने की आवश्यकता नहीं होती। लेकिन जब हम रेस्टोरेंट में पहुंचते हैं, तो हमें बैठने से पहले टेबल के लिए इंतजार करना पड़ सकता है।

पारंपरिक टेलीफोन नेटवर्क सर्किट-स्विच्ड नेटवर्क के उदाहरण हैं। विचार करें कि क्या होता है जब एक व्यक्ति टेलीफोन नेटवर्क पर दूसरे को सूचना (आवाज़ या फैक्स) भेजना चाहता है।

इससे पहले कि प्रेषक सूचना भेज सके, नेटवर्क को दोनों के बीच कनेक्शन स्थापित करना होगा।

प्रेषक और रिसीवर। यह एक वास्तविक कनेक्शन है जिसके लिए प्रेषक और रिसीवर के बीच पथ पर स्विच उस कनेक्शन के लिए कनेक्शन स्थिति बनाए रखते हैं। टेलीफ़ोनी की शब्दावली में, यह

कनेक्शन को सर्किट कहा जाता है। जब नेटवर्क सर्किट स्थापित करता है, तो यह एक स्थिरांक भी सुरक्षित रखता है

कनेक्शन की अवधि के लिए नेटवर्क के लिंक में संचरण दर (प्रत्येक लिंक की संचरण क्षमता के एक अंश का प्रतिनिधित्व करती है)। चूंकि इस प्रेषक-से-

रिसीवर कनेक्शन के माध्यम से, प्रेषक डेटा को रिसीवर तक गारंटीकृत स्थिर दर पर स्थानांतरित कर सकता है।

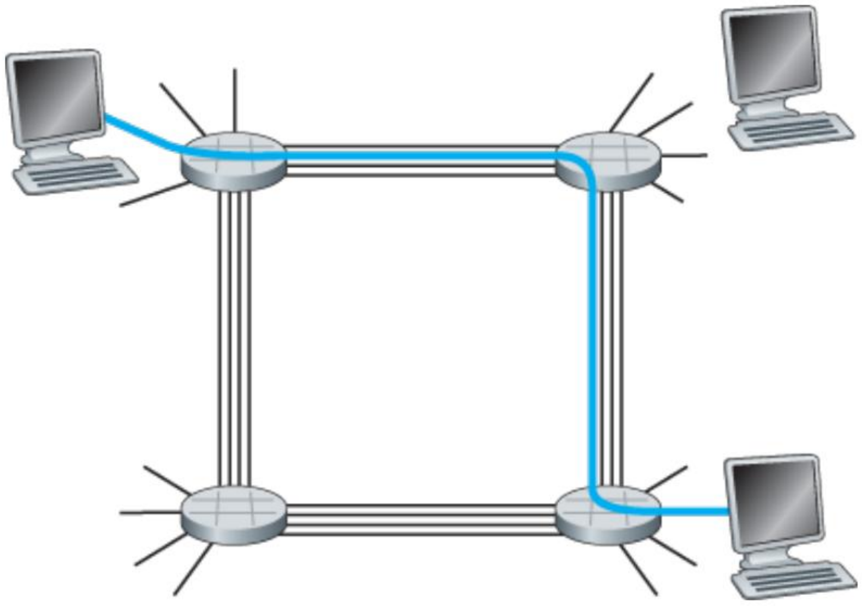
चित्र 1.13 एक सर्किट-स्विच्ड नेटवर्क को दर्शाता है। इस नेटवर्क में, चार सर्किट स्विच चार लिंक द्वारा आपस में जुड़े होते हैं। इनमें से प्रत्येक लिंक में चार सर्किट होते हैं, ताकि प्रत्येक लिंक चार एक साथ कनेक्शन का समर्थन कर सके। होस्ट (उदाहरण के लिए, पीसी और वर्कस्टेशन) प्रत्येक सीधे जुड़े होते हैं

स्विच में से किसी एक को। जब दो होस्ट संवाद करना चाहते हैं, तो नेटवर्क एक समर्पित अंत-

दो होस्ट के बीच टू-एंड कनेक्शन **। इस प्रकार, होस्ट A को होस्ट B के साथ संचार करने के लिए,**

नेटवर्क को पहले दो लिंक में से प्रत्येक पर एक सर्किट आरक्षित करना होगा। इस उदाहरण में, समर्पित एंड-टू-एंड कनेक्शन पहले लिंक में दूसरे सर्किट और दूसरे लिंक में चौथे सर्किट का उपयोग करता है। क्योंकि प्रत्येक

लिंक में चार सर्किट होते हैं, एंड-टू-एंड कनेक्शन द्वारा उपयोग किए जाने वाले प्रत्येक लिंक के लिए, कनेक्शन को कनेक्शन की अवधि के लिए लिंक की कुल ट्रांसमिशन क्षमता का एक चौथाई हिस्सा मिलता है। इस प्रकार, उदाहरण के लिए, यदि आसन्न स्विच के बीच प्रत्येक लिंक की ट्रांसमिशन दर 1 एमबीपीएस है, तो प्रत्येक एंड-टू-एंड सर्किट-स्विच कनेक्शन को 250 केबीपीएस की समर्पित ट्रांसमिशन दर मिलती है।



चित्र 1.13 एक सरल सर्किट-स्विच्ड नेटवर्क जिसमें चार स्विच और चार लिंक शामिल हैं

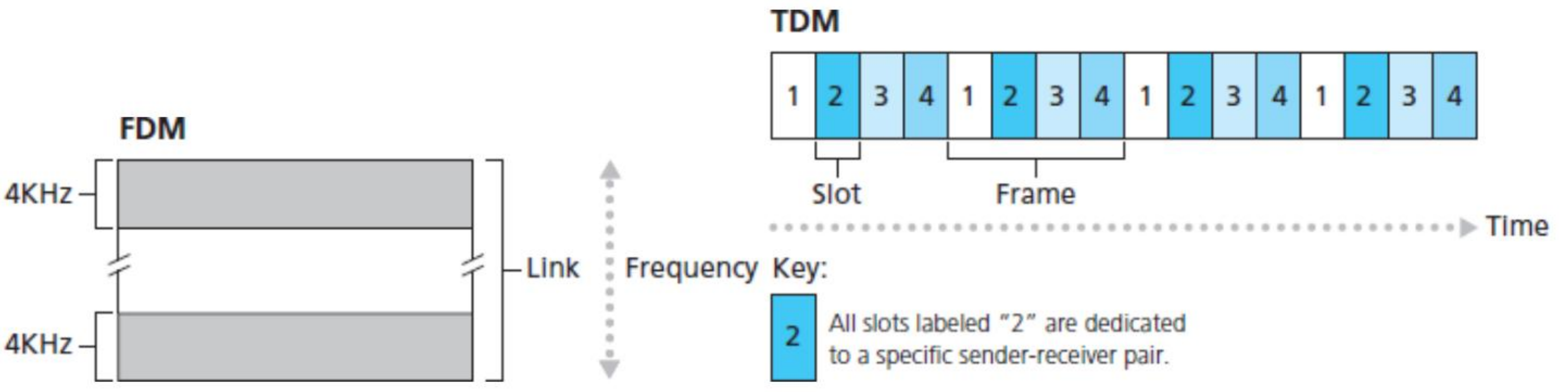
इसके विपरीत, विचार करें कि क्या होता है जब एक होस्ट किसी पैकेट को किसी अन्य होस्ट को पैकेट-स्विच्ड नेटवर्क, जैसे कि इंटरनेट पर भेजना चाहता है। सर्किट स्विचिंग की तरह, पैकेट को संचार लिंक की एक श्रृंखला पर प्रसारित किया जाता है। लेकिन सर्किट स्विचिंग से अलग, पैकेट को किसी भी लिंक संसाधन को आरक्षित किए बिना नेटवर्क में भेजा जाता है। यदि लिंक में से एक भीड़भाड़ वाला है क्योंकि अन्य पैकेट को उसी समय लिंक पर प्रसारित करने की आवश्यकता है, तो पैकेट को ट्रांसमिशन लिंक के भेजने वाले पक्ष में बफर में इंतजार करना होगा और देरी का सामना करना पड़ेगा। इंटरनेट समय पर पैकेट वितरित करने के लिए अपना सर्वश्रेष्ठ प्रयास करता है, लेकिन यह कोई गारंटी नहीं देता है।

सर्किट-स्विच्ड नेटवर्क में मल्टीप्लेक्सिंग

लिंक में सर्किट को या तो फ्रीक्वेंसी-डिवीजन मल्टीप्लेक्सिंग (FDM) या टाइम-डिवीजन मल्टीप्लेक्सिंग (TDM) के साथ लागू किया जाता है। FDM के साथ, लिंक के फ़्रीक्वेंसी स्पेक्ट्रम को लिंक में स्थापित कनेक्शनों के बीच विभाजित किया जाता है। विशेष रूप से, लिंक कनेक्शन की अवधि के लिए प्रत्येक कनेक्शन को एक फ़्रीक्वेंसी बैंड समर्पित करता है। टेलीफ़ोन नेटवर्क में, इस फ़्रीक्वेंसी बैंड की चौड़ाई आम तौर पर 4 kHz (यानी 4,000 हर्ट्ज़ या 4,000 चक्र प्रति सेकंड) होती है। बैंड की चौड़ाई को, आश्चर्यजनक रूप से, कहा जाता है

बैंडविड्थ। एफएम रेडियो स्टेशन 88 मेगाहर्ट्ज और 108 मेगाहर्ट्ज के बीच आवृत्ति स्पेक्ट्रम को साझा करने के लिए एफडीएम का भी उपयोग करते हैं, जिसमें प्रत्येक स्टेशन को एक विशिष्ट आवृत्ति बैंड आवंटित किया जाता है।

टीडीएम लिंक के लिए, समय को निश्चित अवधि के फ़्रेम में विभाजित किया जाता है, और प्रत्येक फ़्रेम को निश्चित संख्या में टाइम स्लॉट में विभाजित किया जाता है। जब नेटवर्क किसी लिंक पर कनेक्शन स्थापित करता है, तो नेटवर्क हर फ़्रेम में एक टाइम स्लॉट इस कनेक्शन को समर्पित करता है। ये स्लॉट उस कनेक्शन के एकमात्र उपयोग के लिए समर्पित होते हैं, जिसमें कनेक्शन के डेटा को संचारित करने के लिए उपयोग के लिए (हर फ़्रेम में) एक टाइम स्लॉट उपलब्ध होता है।



चित्र 1.14 FDM के

साथ, प्रत्येक सर्किट को लगातार बैंडविड्थ का एक अंश मिलता है। TDM के साथ, प्रत्येक सर्किट को समय के संक्षिप्त अंतराल (यानी, स्लॉट के दौरान) के दौरान समय-समय पर सभी बैंडविड्थ मिलती है।

चित्र 1.14 एक विशिष्ट नेटवर्क लिंक के लिए FDM और TDM को दर्शाता है जो अधिकतम चार सर्किट का समर्थन करता है। FDM के लिए, आवृत्ति डोमेन को चार बैंड में विभाजित किया जाता है, जिनमें से प्रत्येक की बैंडविड्थ 4 kHz होती है। TDM के लिए, समय डोमेन को फ्रेम में विभाजित किया जाता है, जिसमें प्रत्येक फ्रेम में चार समय स्लॉट होते हैं; प्रत्येक सर्किट को घूमते हुए TDM फ्रेम में एक ही समर्पित स्लॉट दिया जाता है। TDM के लिए, सर्किट की ट्रांसमिशन दर स्लॉट में बिट्स की संख्या से गुणा की गई फ्रेम दर के बराबर होती है। उदाहरण के लिए, यदि लिंक प्रति सेकंड 8,000 फ्रेम संचारित करता है और प्रत्येक स्लॉट में 8 बिट होते हैं, तो प्रत्येक सर्किट की ट्रांसमिशन दर 64 kbps होती है।

पैकेट स्विचिंग के समर्थकों ने हमेशा तर्क दिया है कि सर्किट स्विचिंग बेकार है क्योंकि समर्पित सर्किट मौन अवधि के दौरान निष्क्रिय रहते हैं। उदाहरण के लिए, जब एक टेलीफोन कॉल में एक व्यक्ति बात करना बंद कर देता है, तो निष्क्रिय नेटवर्क संसाधन (कनेक्शन के मार्ग के साथ लिंक में आवृत्ति बैंड या समय स्लॉट) अन्य चालू कनेक्शन द्वारा उपयोग नहीं किए जा सकते हैं। इन संसाधनों का कैसे कम उपयोग किया जा सकता है, इसके एक अन्य उदाहरण के रूप में, एक रेडियोलॉजिस्ट पर विचार करें जो एक्स-रे की एक श्रृंखला को दूरस्थ रूप से एक्सेस करने के लिए सर्किट-स्विच्ड नेटवर्क का उपयोग करता है। रेडियोलॉजिस्ट एक कनेक्शन स्थापित करता है, एक छवि का अनुरोध करता है, छवि पर विचार करता है, और फिर एक नई छवि का अनुरोध करता है। नेटवर्क संसाधन कनेक्शन को आवंटित किए जाते हैं लेकिन रेडियोलॉजिस्ट के चिंतन अवधि के दौरान उपयोग नहीं किए जाते हैं (यानी, बर्बाद हो जाते हैं)। पैकेट स्विचिंग के समर्थक यह भी बताते हैं कि एंड-टू-एंड सर्किट स्थापित करना और एंड-टू-एंड ट्रांसमिशन क्षमता को आरक्षित करना जटिल है और एंड-टू-एंड पथ के साथ स्विच के संचालन को समन्वित करने के लिए जटिल सिग्नलिंग सॉफ़्टवेयर की आवश्यकता होती है।

सर्किट स्विचिंग पर अपनी चर्चा समाप्त करने से पहले, आइए एक संख्यात्मक उदाहरण के माध्यम से काम करें जो इस विषय पर और अधिक जानकारी प्रदान करेगा। आइए विचार करें कि सर्किट-स्विच्ड नेटवर्क पर होस्ट A से होस्ट B तक 640,000 बिट्स की फ़ाइल भेजने में कितना समय लगता है। मान लीजिए कि नेटवर्क में सभी लिंक 24 स्लॉट के साथ TDM का उपयोग करते हैं और उनकी बिट दर 1.536 Mbps है। यह भी मान लें कि होस्ट A द्वारा फ़ाइल संचारित करने से पहले एंड-टू-एंड सर्किट स्थापित करने में 500 msec लगते हैं। फ़ाइल भेजने में कितना समय लगता है? प्रत्येक सर्किट

इसकी संचरण दर (1.536 एमबीपीएस)/24=64 केबीपीएस है, इसलिए इसे (640,000 बिट्स)/(64 केबीपीएस)=10 सेकंड लगते हैं

फ़ाइल को प्रेषित करें। इस 10 सेकंड में हम सर्किट स्थापना समय जोड़ते हैं, जिससे फ़ाइल भेजने के लिए 10.5 सेकंड मिलते हैं। ध्यान दें कि ट्रांसमिशन समय लिंक की संख्या से स्वतंत्र है: यदि एंड-टू-एंड सर्किट एक लिंक या सौ लिंक से होकर गुज़रता है तो ट्रांसमिशन समय 10 सेकंड होगा। (वास्तविक

अंत-से-अंत विलंब में प्रसार विलंब भी शामिल है; अनुभाग 1.4 देखें।)

पैकेट स्विचिंग बनाम सर्किट स्विचिंग

सर्किट स्विचिंग और पैकेट स्विचिंग का वर्णन करने के बाद, आइए दोनों की तुलना करें। पैकेट स्विचिंग के आलोचकों ने अक्सर तर्क दिया है कि पैकेट स्विचिंग वास्तविक समय की सेवाओं (उदाहरण के लिए, टेलीफोन कॉल और वीडियो कॉन्फ्रेंस कॉल) के लिए उपयुक्त नहीं है क्योंकि इसमें परिवर्तनशील और अप्रत्याशित अंत-से-अंत देरी होती है (मुख्य रूप से परिवर्तनशील और अप्रत्याशित कतारबद्ध देरी के कारण)। पैकेट स्विचिंग के समर्थकों का तर्क है कि (1) यह सर्किट स्विचिंग की तुलना में ट्रांसमिशन क्षमता का बेहतर साझाकरण प्रदान करता है और (2) यह सर्किट स्विचिंग की तुलना में लागू करने के लिए सरल, अधिक कुशल और कम खर्चीला है। पैकेट स्विचिंग की एक दिलचस्प चर्चा

स्विचिंग बनाम सर्किट स्विचिंग [मोलिनेरो-फर्नांडीज 2002] है। आम तौर पर, जो लोग रेस्तरां आरक्षण के झंझट में नहीं पड़ना चाहते, वे सर्किट स्विचिंग के बजाय पैकेट स्विचिंग को प्राथमिकता देते हैं।

पैकेट स्विचिंग अधिक कुशल क्यों है? आइए एक सरल उदाहरण देखें। मान लीजिए कि उपयोगकर्ता 1 एमबीपीएस लिंक साझा करते हैं। यह भी मान लें कि प्रत्येक उपयोगकर्ता गतिविधि की अवधि के बीच बारी-बारी से आता है, जब उपयोगकर्ता 100 केबीपीएस की स्थिर दर पर डेटा उत्पन्न करता है, और निष्क्रियता की अवधि, जब उपयोगकर्ता कोई डेटा उत्पन्न नहीं करता है। मान लीजिए कि उपयोगकर्ता केवल 10 प्रतिशत समय सक्रिय रहता है (और शेष 90 प्रतिशत समय के दौरान आराम से कॉफी पीता रहता है)

समय का प्रतिशत)। सर्किट स्विचिंग के साथ, **हर समय** प्रत्येक **उपयोगकर्ता** के लिए **100 केबीपीएस** आरक्षित **होना चाहिए** । उदाहरण के लिए, सर्किट-स्विच्ड टीडीएम के साथ, यदि एक सेकंड के फ्रेम को 100 एमएस के 10 टाइम स्लॉट में विभाजित किया जाता है, तो प्रत्येक उपयोगकर्ता को प्रति फ्रेम एक टाइम स्लॉट आवंटित किया जाएगा।

इस प्रकार, सर्किट-स्विच्ड लिंक केवल 10(=1 एमबीपीएस/100 केबीपीएस)एक साथ उपयोगकर्ताओं का समर्थन कर सकता है। पैकेट स्विचिंग के साथ, एक विशिष्ट उपयोगकर्ता के सक्रिय होने की संभावना 0.1 (यानी, 10 प्रतिशत) है। यदि 35 उपयोगकर्ता हैं, तो 11 या उससे अधिक एक साथ सक्रिय उपयोगकर्ताओं की संभावना लगभग 0.0004 है।

(होमवर्क समस्या P8 बताती है कि यह संभावना कैसे प्राप्त की जाती है।) जब 10 या उससे कम सक्रिय उपयोगकर्ता एक साथ होते हैं (जो कि 0.9996 संभावना के साथ होता है), तो डेटा की कुल आगमन दर 1 एमबीपीएस से कम या उसके बराबर होती है, जो लिंक की आउटपुट दर है। इस प्रकार, जब 10 या उससे कम सक्रिय उपयोगकर्ता होते हैं, तो उपयोगकर्ताओं के पैकेट अनिवार्य रूप से बिना किसी देरी के लिंक के माध्यम से प्रवाहित होते हैं, जैसा कि सर्किट स्विचिंग के मामले में होता है। जब 10 से अधिक सक्रिय उपयोगकर्ता एक साथ होते हैं, तो पैकेट की कुल आगमन दर लिंक की आउटपुट क्षमता से अधिक हो जाती है, और आउटपुट कतार बढ़ने लगेगी। (यह तब तक बढ़ता रहता है जब तक कि कुल इनपुट दर 1 एमबीपीएस से नीचे नहीं आ जाती, जिस बिंदु पर कतार की लंबाई कम होने लगेगी।)

क्योंकि इस ऐप में एक साथ 10 से अधिक सक्रिय उपयोगकर्ता होने की संभावना नगण्य है।

उदाहरण के लिए, पैकेट स्विचिंग मूलतः सर्किट स्विचिंग के समान ही प्रदर्शन प्रदान करती है, लेकिन ऐसा करते समय यह उपयोगकर्ताओं की संख्या से तीन गुना अधिक उपयोगकर्ताओं को अनुमति देती है।

आइए अब एक दूसरे सरल उदाहरण पर विचार करें। मान लीजिए कि 10 उपयोगकर्ता हैं और उनमें से एक उपयोगकर्ता अचानक एक हज़ार 1,000-बिट पैकेट उत्पन्न करता है, जबकि अन्य उपयोगकर्ता निष्क्रिय रहते हैं और पैकेट उत्पन्न नहीं करते हैं। TDM सर्किट स्विचिंग के तहत 10 स्लॉट प्रति फ़्रेम और प्रत्येक स्लॉट में 1,000 बिट्स होते हैं, सक्रिय उपयोगकर्ता डेटा संचारित करने के लिए प्रति फ़्रेम केवल अपने एक टाइम स्लॉट का उपयोग कर सकता है, जबकि शेष नौ टाइम स्लॉट

प्रत्येक फ्रेम में निष्क्रिय रहते हैं। सक्रिय उपयोगकर्ता के सभी दस लाख बिट्स डेटा के संग्रहित होने में 10 सेकंड का समय लगेगा

प्रेषित किया गया है। पैकेट स्विचिंग के मामले में, सक्रिय उपयोगकर्ता लगातार 1 एमबीपीएस की पूर्ण लिंक दर पर अपने पैकेट भेज सकता है, क्योंकि कोई अन्य उपयोगकर्ता ऐसे पैकेट नहीं बना रहा है जिन्हें सक्रिय उपयोगकर्ता के पैकेट के साथ मल्टीप्लेक्स करने की आवश्यकता है। इस मामले में, सक्रिय उपयोगकर्ता का सारा डेटा प्रेषित किया जाएगा
1 सेकंड के भीतर.

उपरोक्त उदाहरण दो तरीकों को दर्शाते हैं जिसमें पैकेट स्विचिंग का प्रदर्शन सर्किट स्विचिंग से बेहतर हो सकता है। वे कई डेटा स्ट्रीम के बीच लिंक की ट्रांसमिशन दर को साझा करने के दो तरीकों के बीच महत्वपूर्ण अंतर को भी उजागर करते हैं। सर्किट स्विचिंग मांग की परवाह किए बिना ट्रांसमिशन लिंक के उपयोग को पूर्व-आवंटित करता है, जिसमें आवंटित लेकिन अनावश्यक लिंक समय अप्रयुक्त रहता है। पैकेट

दूसरी ओर स्विचिंग ऑन करने से मांग के अनुसार लिंक का उपयोग आवंटित होता है। लिंक ट्रांसमिशन क्षमता पैकेट-दर-पैकेट आधार पर केवल उन उपयोगकर्ताओं के बीच साझा की जाएगी जिनके पास पैकेट हैं जिन्हें प्रसारित करने की आवश्यकता है
लिंक.

हालाँकि पैकेट स्विचिंग और सर्किट स्विचिंग दोनों ही आज के दूरसंचार में प्रचलित हैं
नेटवर्क में, प्रवृत्ति निश्चित रूप से पैकेट स्विचिंग की दिशा में रही है। आज के कई सर्किट-
स्विच किए गए टेलीफ़ोन नेटवर्क धीरे-धीरे पैकेट स्विचिंग की ओर बढ़ रहे हैं। विशेष रूप से, टेलीफ़ोन
नेटवर्क अक्सर टेलीफोन कॉल के महंगे विदेशी हिस्से के लिए पैकेट स्विचिंग का उपयोग करते हैं।

1.3.3 नेटवर्कों का नेटवर्क

हमने पहले देखा कि एंड सिस्टम (पीसी, स्मार्टफोन, वेब सर्वर, मेल सर्वर, इत्यादि) एक्सेस ISP के माध्यम से इंटरनेट से जुड़ते हैं। एक्सेस ISP DSL, केबल, FTTH, वाई-फाई और सेलुलर सहित एक्सेस तकनीकों की एक सरणी का उपयोग करके वायर्ड या वायरलेस कनेक्टिविटी प्रदान कर सकता है। ध्यान दें कि एक्सेस ISP को टेल्को या केबल कंपनी होने की आवश्यकता नहीं है; इसके बजाय यह, उदाहरण के लिए, एक विश्वविद्यालय (छात्रों, कर्मचारियों और संकाय को इंटरनेट एक्सेस प्रदान करना) या एक कंपनी (अपने कर्मचारियों के लिए एक्सेस प्रदान करना) हो सकता है। लेकिन एंड यूजर्स और कंटेंट प्रोवाइडर्स को एक्सेस ISP से जोड़ना, इंटरनेट को बनाने वाले अरबों एंड सिस्टम को जोड़ने की पहेली को सुलझाने का एक छोटा सा हिस्सा है। इसे पूरा करने के लिए

पहेली, एक्सेस आईएसपी को खुद आपस में जुड़े रहना चाहिए। यह **नेटवर्क** का एक नेटवर्क **बनाकर किया जाता है** - इस वाक्यांश को समझना इंटरनेट को समझने की कुंजी है।

पिछले कुछ वर्षों में, नेटवर्कों का वह नेटवर्क जो इंटरनेट का निर्माण करता है, एक बहुत ही जटिल संरचना में विकसित हो गया है। इस विकास का अधिकांश हिस्सा प्रदर्शन के बजाय अर्थशास्त्र और राष्ट्रीय नीति द्वारा संचालित होता है
आज के इंटरनेट नेटवर्क संरचना को समझने के लिए, आइए क्रमिक रूप से एक संरचना का निर्माण करें।
नेटवर्क संरचनाओं की श्रृंखला, जिसमें प्रत्येक नई संरचना आज हमारे पास मौजूद जटिल इंटरनेट का बेहतर अनुमान है। याद रखें कि व्यापक लक्ष्य एक्सेस आईएसपी को आपस में जोड़ना है ताकि सभी अंतिम सिस्टम एक दूसरे को पैकेट भेज सकें। एक सरल दृष्टिकोण यह होगा कि प्रत्येक एक्सेस को

आईएसपी **हर दूसरे एक्सेस आईएसपी से** सीधे जुड़ता है। इस तरह का मेश डिज़ाइन, ज़ाहिर है, एक्सेस आईएसपी के लिए बहुत महंगा है, क्योंकि इसके लिए प्रत्येक एक्सेस आईएसपी को प्रत्येक के लिए एक अलग संचार लिंक की आवश्यकता होगी
दुनिया भर में लाखों अन्य आईएसपी तक पहुंच है।

हमारी पहली नेटवर्क संरचना, नेटवर्क संरचना 1, सभी एक्सेस आईएसपी को एक वैश्विक ट्रांजिट आईएसपी के साथ जोड़ती है। हमारा (काल्पनिक) वैश्विक ट्रांजिट आईएसपी राउटर और संचार लिंक का एक नेटवर्क है जो न केवल दुनिया भर में फैला है, बल्कि सैकड़ों हजारों एक्सेस आईएसपी में से प्रत्येक के पास कम से कम एक राउटर भी है। बेशक, वैश्विक आईएसपी के लिए इतना व्यापक नेटवर्क बनाना बहुत महंगा होगा। लाभदायक होने के लिए, यह स्वाभाविक रूप से कनेक्टिविटी के लिए प्रत्येक एक्सेस आईएसपी से शुल्क लेगा, जिसमें मूल्य निर्धारण एक एक्सेस आईएसपी द्वारा वैश्विक आईएसपी के साथ आदान-प्रदान किए जाने वाले ट्रैफ़िक की मात्रा को दर्शाता है (लेकिन जरूरी नहीं कि सीधे आनुपातिक हो)। चूंकि एक्सेस आईएसपी वैश्विक ट्रांजिट आईएसपी को भुगतान करता है ,

अब यदि कोई कंपनी एक वैश्विक ट्रांजिट आईएसपी का निर्माण और संचालन करती है जो लाभदायक है, तो यह स्वाभाविक है कि अन्य कंपनियां अपने स्वयं के वैश्विक ट्रांजिट आईएसपी का निर्माण करें और मूल वैश्विक ट्रांजिट आईएसपी के साथ प्रतिस्पर्धा करें।

इससे नेटवर्क संरचना 2 की स्थापना होती है, जिसमें हजारों की संख्या में एक्सेस आईएसपी और नेटवर्क ऑपरेटर शामिल होते हैं।

कई वैश्विक पारगमन ISPs। एक्सेस ISP निश्चित रूप से नेटवर्क संरचना 1 की तुलना में नेटवर्क संरचना 2 को प्राथमिकता देते हैं क्योंकि वे अब अपने मूल्य निर्धारण और सेवाओं के आधार पर प्रतिस्पर्धी वैश्विक पारगमन प्रदाताओं में से चुन सकते हैं। हालाँकि, ध्यान दें कि वैश्विक पारगमन ISPs को स्वयं आपस में जुड़ना चाहिए: अन्यथा वैश्विक पारगमन प्रदाताओं में से किसी एक से जुड़े एक्सेस ISP अन्य वैश्विक पारगमन प्रदाताओं से जुड़े एक्सेस ISPs के साथ संवाद करने में सक्षम नहीं होंगे।

नेटवर्क संरचना 2, जिसका अभी वर्णन किया गया है, एक दो-स्तरीय पदानुक्रम है जिसमें वैश्विक पारगमन प्रदाता शीर्ष स्तर पर और पहुँच ISP निचले स्तर पर रहते हैं। यह मानता है कि वैश्विक पारगमन ISP न केवल प्रत्येक पहुँच ISP के करीब पहुंचने में सक्षम हैं, बल्कि ऐसा करना आर्थिक रूप से वांछनीय भी पाते हैं। हकीकत में, हालांकि कुछ ISP के पास प्रभावशाली वैश्विक कवरेज है और वे कई पहुँच ISP से सीधे जुड़ते हैं, कोई भी ISP दुनिया के प्रत्येक शहर में मौजूद नहीं है। इसके बजाय, किसी भी क्षेत्र में एक क्षेत्रीय ISP **हो सकता है** जिससे उस क्षेत्र के पहुँच ISP जुड़ते हैं। फिर प्रत्येक क्षेत्रीय ISP **टियर** -1 ISP से जुड़ता है। टियर-1 ISP हमारे (काल्पनिक) वैश्विक पारगमन ISP के समान हैं; लेकिन टियर-1 ISP, जो वास्तव में मौजूद हैं, उनकी दुनिया के हर शहर में मौजूदगी नहीं है। जैसा कि कहावत है - यदि आपसे पूछा जाए कि क्या आप किसी समूह के सदस्य हैं, तो संभवतः आप नहीं हैं।

नेटवर्कों के इस नेटवर्क पर लौटते हुए, न केवल कई प्रतिस्पर्धी टियर-1 आईएसपी हैं, बल्कि एक क्षेत्र में कई प्रतिस्पर्धी क्षेत्रीय आईएसपी भी हो सकते हैं। इस तरह के पदानुक्रम में, प्रत्येक एक्सेस आईएसपी उस क्षेत्रीय आईएसपी को भुगतान करता है जिससे वह जुड़ता है, और प्रत्येक क्षेत्रीय आईएसपी उस टियर-1 आईएसपी को भुगतान करता है जिससे वह जुड़ता है। (एक एक्सेस आईएसपी सीधे टियर-1 आईएसपी से भी जुड़ सकता है, ऐसी स्थिति में वह टियर-1 आईएसपी को भुगतान करता है)। इस प्रकार, पदानुक्रम के प्रत्येक स्तर पर ग्राहक-प्रदाता संबंध होता है। ध्यान दें कि टियर-1 आईएसपी किसी को भुगतान नहीं करते क्योंकि वे पदानुक्रम में सबसे ऊपर होते हैं। मामले को और जटिल बनाने के लिए, कुछ क्षेत्रों में एक बड़ा क्षेत्रीय आईएसपी (संभवतः पूरे देश में फैला हुआ) हो सकता है जिससे उस क्षेत्र के छोटे क्षेत्रीय आईएसपी जुड़ते हैं उदाहरण के लिए, चीन में, प्रत्येक शहर में एक्सेस आईएसपी हैं, जो प्रांतीय आईएसपी से जुड़ते हैं, जो बदले में राष्ट्रीय आईएसपी से जुड़ते हैं, जो अंततः एक दूसरे से जुड़ते हैं।

टियर-1 आईएसपी से कनेक्ट करें [तियान 2012]। हम इस बहु-स्तरीय पदानुक्रम का उल्लेख करते हैं, जो अभी भी केवल एक कच्चा है

आज के इंटरनेट का सन्निकटन, नेटवर्क संरचना 3 के रूप में।

आज के इंटरनेट से ज्यादा मिलता-जुलता नेटवर्क बनाने के लिए हमें पदानुक्रमिक नेटवर्क संरचना 3 में उपस्थिति बिंदु (PoPs), मल्टी-होमिंग, पियरिंग और इंटरनेट एक्सचेंज पॉइंट्स (IXPs) जोड़ना होगा। PoPs पदानुक्रम के सभी स्तरों में मौजूद हैं, निचले (पहुंच ISP) स्तर को छोड़कर। एक PoP बस प्रदाता के नेटवर्क में एक या एक से अधिक राउटर (एक ही स्थान पर) का एक समूह है जहां ग्राहक ISP प्रदाता ISP से जुड़ सकते हैं। किसी ग्राहक नेटवर्क को किसी प्रदाता के PoP से कनेक्ट करने के लिए, वह अपने किसी राउटर को PoP पर स्थित राउटर से सीधे जोड़ने के लिए किसी तृतीय-पक्ष दूरसंचार प्रदाता से हाई -स्पीड लिंक पट्टे **पर ले सकता है। कोई भी** ISP (टियर-1 ISP को छोड़कर) मल्टी-होम चुन सकता है इसी तरह, एक क्षेत्रीय ISP कई टियर-1 ISP के साथ मल्टी-होम कर सकता है। जब कोई ISP मल्टी-होम करता है, तो वह इंटरनेट पर पैकेट भेजना और प्राप्त करना जारी रख सकता है, भले ही उसके प्रदाताओं में से कोई एक विफल हो जाए।

जैसा कि हमने अभी सीखा, ग्राहक ISP वैश्विक इंटरनेट इंटरकनेक्टिविटी प्राप्त करने के लिए अपने प्रदाता ISP को भुगतान करते हैं। ग्राहक ISP द्वारा प्रदाता ISP को भुगतान की जाने वाली राशि, प्रदाता के साथ उसके द्वारा आदान-प्रदान किए जाने वाले ट्रैफ़िक की मात्रा को दर्शाती है। इन लागतों को कम करने के लिए, पदानुक्रम के समान स्तर पर पास के ISP की एक जोड़ी सहकर्मी हो सकती है, अर्थात, वे अपने नेटवर्क को सीधे एक साथ जोड़ सकते हैं ताकि उनके बीच का सारा ट्रैफ़िक अपस्ट्रीम बिचौलियों के बजाय सीधे कनेक्शन से गुज़रे। जब दो ISP सहकर्मी होते हैं, तो यह आम तौर पर निपटान-मुक्त होता है, अर्थात, कोई भी ISP दूसरे को भुगतान नहीं करता है। जैसा कि पहले उल्लेख किया गया है, टियर-1 ISP भी एक-दूसरे के साथ सहकर्मी होते हैं, निपटान-मुक्त। सहकर्मी और ग्राहक-प्रदाता संबंधों की एक पठनीय चर्चा के लिए, [वैन डेर बर्ग 2008] देखें। इसी तरह, एक तृतीय-पक्ष कंपनी एक इंटरनेट एक्सचेंज पॉइंट (IXP) बना सकती है, जो एक मिलन बिंदु है जहाँ कई ISP एक साथ सहकर्मी हो सकते हैं। एक IXP आम तौर पर एक में होता है

अपने स्वयं के स्विच के साथ स्टैंड-अलोन बिल्डिंग [एगर 2012]। आज इंटरनेट पर 400 से अधिक IXP हैं [आईएक्सपी सूची 2016]। **हम इस पारिस्थितिकी तंत्र को** नेटवर्क संरचना 4 के रूप में संदर्भित करते हैं - जिसमें एक्सेस आईएसपी, क्षेत्रीय आईएसपी, टियर-1 आईएसपी, पीओपी, मल्टी-होमिंग, पीयरिंग और आईएक्सपी शामिल हैं।

अब हम अंततः नेटवर्क संरचना 5 पर पहुँचते हैं, जो आज के इंटरनेट का वर्णन करता है। नेटवर्क संरचना 5, चित्र 1.15 में दर्शाया गया है, सामग्री-प्रदाता नेटवर्क को जोड़कर नेटवर्क संरचना 4 के शीर्ष पर निर्मित होता है ।

Google वर्तमान में ऐसे कंटेंट प्रदाता नेटवर्क के अग्रणी उदाहरणों में से एक है। इस लेखन के अनुसार, यह अनुमान लगाया गया है कि Google के उत्तरी अमेरिका, यूरोप, एशिया में 50-100 डेटा सेंटर हैं।

दक्षिण अमेरिका और ऑस्ट्रेलिया। इनमें से कुछ डेटा सेंटर में एक लाख से ज़्यादा सर्वर हैं,

जबकि अन्य डेटा सेंटर छोटे हैं, जिनमें केवल सैकड़ों सर्वर हैं। Google डेटा सेंटर सभी Google के निजी TCP/IP नेटवर्क के माध्यम से आपस में जुड़े हुए हैं, जो पूरी दुनिया में फैला हुआ है, लेकिन फिर भी सार्वजनिक इंटरनेट से अलग है। महत्वपूर्ण बात यह है कि Google निजी नेटवर्क केवल ट्रैफ़िक को ले जाता है

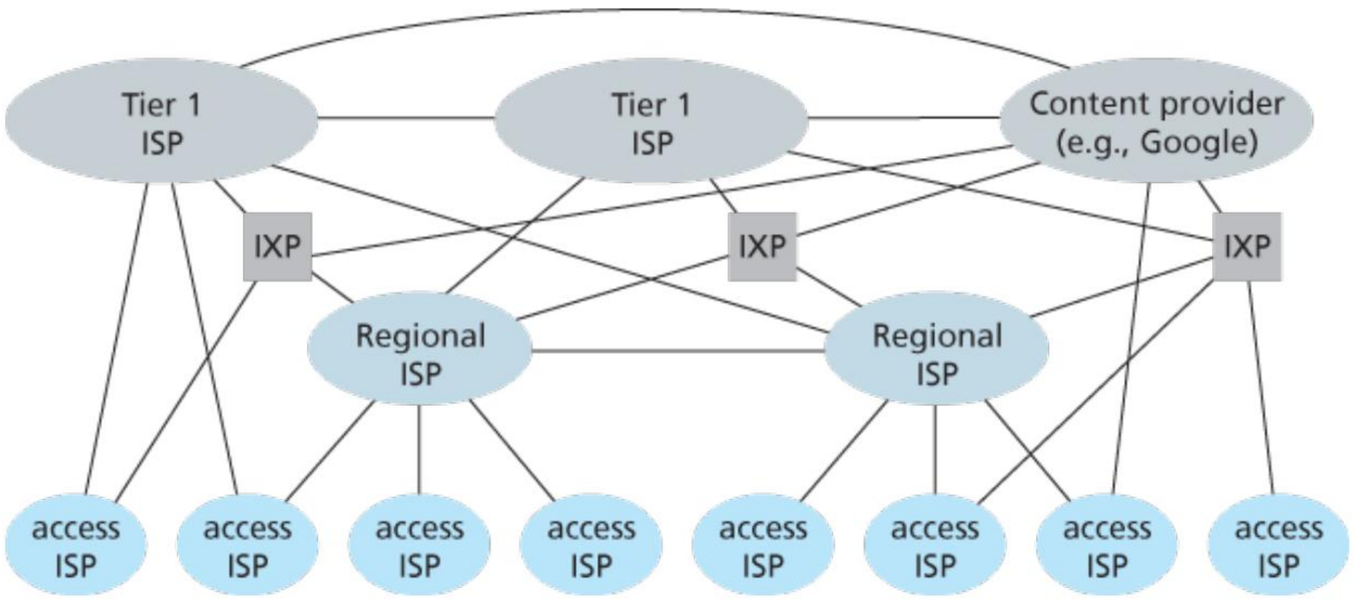
गूगल सर्वर। जैसा कि चित्र 1.15 में दिखाया गया है, गूगल निजी नेटवर्क निचले स्तर के आईएसपी के साथ सीधे जुड़कर (निपटान मुक्त) इंटरनेट के ऊपरी स्तरों को "बाईपास" करने का प्रयास करता है

उन्हें या IXPs पर उनके साथ जुड़कर [लैबोविट्ज़ 2010]। हालाँकि, चूँकि कई एक्सेस ISP तक अभी भी केवल टियर-1 नेटवर्क के माध्यम से ही पहुँचा जा सकता है, इसलिए Google नेटवर्क टियर-1 ISP से भी जुड़ता है, और उन ISP को उनके साथ आदान-प्रदान किए जाने वाले ट्रैफ़िक के लिए भुगतान करता है। अपना खुद का नेटवर्क बनाकर, एक कंटेंट

प्रदाता न केवल उच्च-स्तरीय आईएसपी को अपने भुगतान को कम करता है, बल्कि अंतिम उपयोगकर्ताओं तक अपनी सेवाओं को कैसे पहुंचाया जाए, इस पर भी उसका अधिक नियंत्रण होता है। Google के नेटवर्क इंफ्रास्ट्रक्चर का अधिक विस्तृत विवरण नीचे दिया गया है

अनुभाग 2.6 में विस्तार से जानकारी दी गई है ।

संक्षेप में, आज का इंटरनेट - नेटवर्कों का एक नेटवर्क - जटिल है, जिसमें एक दर्जन या उससे अधिक टियर-1 आईएसपी और सैकड़ों हज़ारों निचले-स्तरीय आईएसपी शामिल हैं। आईएसपी अपने कवरेज में विविधता रखते हैं, जिनमें से कुछ कई महाद्वीपों और महासागरों में फैले हुए हैं, और अन्य संकीर्ण भौगोलिक क्षेत्रों तक सीमित हैं। टियर ISP उच्च-स्तरीय ISP से जुड़ते हैं, और उच्च-स्तरीय ISP एक-दूसरे से जुड़ते हैं। उपयोगकर्ता और सामग्री प्रदाता निम्न-स्तरीय ISP के ग्राहक हैं, और निम्न-स्तरीय ISP उच्च-स्तरीय ISP के ग्राहक हैं। हाल के वर्षों में, प्रमुख सामग्री प्रदाताओं ने भी अपने स्वयं के नेटवर्क बनाए हैं और जहाँ संभव हो, सीधे निम्न-स्तरीय ISP से जुड़ते हैं।



चित्र 1.15 आईएसपी का अंतर्संबंध

1.4 पैकेट-स्विच्ड नेटवर्क में विलंब, हानि और थ्रूपुट

खंड 1.1 में हमने कहा था कि इंटरनेट को एक ऐसे बुनियादी ढांचे के रूप में देखा जा सकता है जो एंड सिस्टम पर चलने वाले वितरित अनुप्रयोगों को सेवाएँ प्रदान करता है। आदर्श रूप से, हम चाहेंगे कि इंटरनेट सेवाएँ किसी भी दो एंड सिस्टम के बीच जितना चाहें उतना डेटा स्थानांतरित करने में सक्षम हों, बिना किसी डेटा हानि के। अफसोस, यह एक बड़ा लक्ष्य है, जो वास्तविकता में अप्राप्य है। इसके बजाय, कंप्यूटर नेटवर्क आवश्यक रूप से एंड सिस्टम के बीच थ्रूपुट (प्रति सेकंड स्थानांतरित किए जा सकने वाले डेटा की मात्रा) को बाधित करते हैं, एंड सिस्टम के बीच देरी का परिचय देते हैं, और वास्तव में पैकेट खो सकते हैं। एक ओर, यह दुर्भाग्यपूर्ण है कि वास्तविकता के भौतिक नियम देरी और हानि के साथ-साथ थ्रूपुट को भी बाधित करते हैं। दूसरी ओर, क्योंकि कंप्यूटर नेटवर्क में ये समस्याएँ हैं, इसलिए समस्याओं से निपटने के तरीके के बारे में कई आकर्षक मुद्दे हैं - कंप्यूटर नेटवर्किंग पर एक कोर्स भरने और हजारों पीएचडी थीसिस को प्रेरित करने के लिए पर्याप्त से अधिक मुद्दे! इस खंड में, हम कंप्यूटर नेटवर्क में देरी, हानि और थ्रूपुट की जांच और मात्रा निर्धारित करना शुरू करेंगे।

1.4.1 पैकेट-स्विच्ड नेटवर्क में विलंब का अवलोकन

याद रखें कि एक पैकेट एक होस्ट (स्रोत) से शुरू होता है, राउटर की एक श्रृंखला से गुजरता है, और दूसरे होस्ट (गंतव्य) में अपनी यात्रा समाप्त करता है। जब एक पैकेट एक नोड (होस्ट या राउटर) से दूसरे नोड (स्रोत) तक जाता है, तो यह राउटर की एक श्रृंखला से होकर गुजरता है, और दूसरे होस्ट (गंतव्य) में अपनी यात्रा समाप्त करता है।

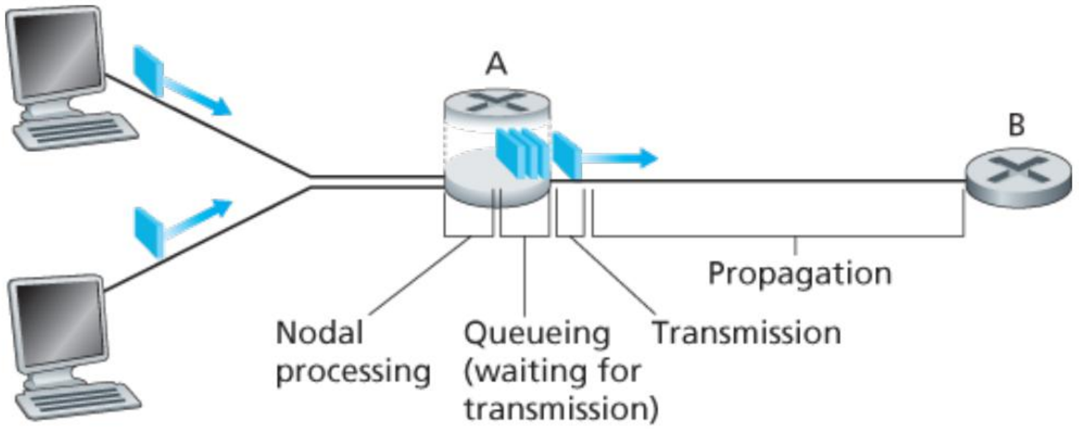
इस पथ पर आने वाले प्रत्येक नोड (होस्ट या राउटर) पर, पैकेट को पथ पर प्रत्येक **नोड पर कई प्रकार की देरी का सामना करना पड़ता है। इनमें से सबसे महत्वपूर्ण देरी** नोडल प्रोसेसिंग देरी, कतार में देरी, ट्रांसमिशन देरी और प्रसार देरी है; साथ में, ये देरी कुल नोडल देरी देने के लिए एकत्रित होती हैं। कई इंटरनेट अनुप्रयोगों का प्रदर्शन - जैसे खोज, वेब ब्राउज़िंग, ई-मेल, मानचित्र, त्वरित संदेश और वॉयस-ओवर-आईपी - नेटवर्क देरी से बहुत प्रभावित होते हैं। पैकेट स्विचिंग और कंप्यूटर नेटवर्क की गहरी समझ हासिल करने के लिए, हमें इन देरी की प्रकृति और महत्व को समझना चाहिए।

विलंब के प्रकार

आइए चित्र 1.16 के संदर्भ में इन देरी का पता लगाएं। स्रोत और गंतव्य के बीच अपने अंत-से-अंत मार्ग के हिस्से के रूप में, राउटर ए के माध्यम से अपस्ट्रीम नोड से राउटर बी तक एक पैकेट भेजा जाता है। हमारा लक्ष्य राउटर ए पर नोडल देरी को चिह्नित करना है। ध्यान दें कि राउटर ए में राउटर बी तक जाने वाला एक आउटबाउंड लिंक है।

इस लिंक से पहले एक कतार होती है (जिसे बफर भी कहते हैं)। जब पैकेट अपस्ट्रीम नोड से राउटर A पर आता है, तो राउटर A पैकेट के हेडर की जांच करता है ताकि पैकेट के लिए उपयुक्त आउटबाउंड लिंक निर्धारित किया जा सके और फिर पैकेट को इस लिंक पर निर्देशित किया जा सके। इस उदाहरण में, पैकेट के लिए आउटबाउंड लिंक वह है जो राउटर B की ओर जाता है। किसी पैकेट को लिंक पर तभी प्रसारित किया जा सकता है जब वर्तमान में कोई अन्य पैकेट न हो

लिंक पर प्रेषित किया जा रहा है और यदि कतार में इससे पहले कोई अन्य पैकेट नहीं हैं; यदि लिंक



चित्र 1.16 राउटर A पर नोडल विलंब

यदि लिंक वर्तमान में व्यस्त है या लिंक के लिए पहले से ही अन्य पैकेट कतार में हैं, तो नया आने वाला पैकेट कतार में शामिल हो जाएगा।

प्रसंस्करण में देरी

पैकेट के हेडर की जांच करने और पैकेट को कहां निर्देशित करना है, यह निर्धारित करने में लगने वाला समय प्रोसेसिंग देरी का हिस्सा है। प्रोसेसिंग देरी में अन्य कारक भी शामिल हो सकते हैं, जैसे पैकेट में बिट-लेवल त्रुटियों की जांच करने के लिए आवश्यक समय जो अपस्ट्रीम नोड से राउटर ए तक पैकेट के बिट्स को संचारित करने में हुआ। उच्च गति वाले राउटर में प्रोसेसिंग देरी आमतौर पर माइक्रोसेकंड या उससे कम के क्रम की होती है। इस नोडल प्रोसेसिंग के बाद, राउटर पैकेट को उस कतार में निर्देशित करता है जो लिंक से पहले होती है

राउटर बी. (अध्याय 4 **में** हम विस्तार से अध्ययन करेंगे कि राउटर कैसे काम करता है।)

कतार में देरी

कतार में, पैकेट को कतार में देरी का **अनुभव होता है** क्योंकि यह लिंक पर प्रसारित होने की प्रतीक्षा करता है। किसी विशिष्ट पैकेट की कतार में देरी की अवधि पहले आने वाले पैकेटों की संख्या पर निर्भर करेगी जो कतार में हैं और लिंक पर प्रसारण के लिए प्रतीक्षा कर रहे हैं। यदि कतार खाली है और वर्तमान में कोई अन्य पैकेट प्रसारित नहीं हो रहा है, तो हमारे पैकेट की कतार में देरी शून्य होगी। दूसरी ओर, यदि ट्रैफ़िक भारी है और कई अन्य पैकेट भी प्रसारित होने की प्रतीक्षा कर रहे हैं, तो कतार में देरी लंबी होगी। हम जल्द ही देखेंगे कि आने वाले पैकेट को मिलने वाले पैकेटों की संख्या कतार में आने वाले ट्रैफ़िक की तीव्रता और प्रकृति का एक कार्य है। व्यवहार में कतार में देरी माइक्रोसेकंड से लेकर मिलीसेकंड तक हो सकती है।

संचरण विलंब

यह मानते हुए कि पैकेट पहले आओ पहले पाओ के आधार पर प्रेषित किए जाते हैं, जैसा कि पैकेट-स्विच्ड नेटवर्क में आम है, हमारा पैकेट तभी प्रेषित किया जा सकता है जब उससे पहले आए सभी पैकेट्स को भेज दिया जाए।

प्रेषित किए गए हैं। पैकेट की लंबाई को L बिट्स से निरूपित करें, और संचरण दर को निरूपित करें

राउटर A से राउटर B तक R **बिट्स/सेकंड की** दर से लिंक । उदाहरण के लिए, 10 एमबीपीएस ईथरनेट लिंक के लिए, दर R=10 एमबीपीएस है; R=100 एमबीपीएस। ईथरनेट लिंक के लिए है, यह दर पैकेट के सभी बिट्स को ट्रांसमिशन देरी एल/आर है । यह 100 एमबीपीएस लिंक में धकेलने (यानी, संचारित करने) के लिए आवश्यक समय की मात्रा है।

व्यावहारिक रूप से संचरण में विलंब आमतौर पर माइक्रोसेकंड से लेकर मिलीसेकंड तक होता है।

प्रचार देरी

एक बार जब बिट को लिंक में धकेल दिया जाता है, तो उसे राउटर बी तक प्रसारित करने की आवश्यकता होती है। लिंक की शुरुआत से राउटर बी तक प्रसारित होने में लगने वाला समय प्रसार विलंब है। बिट लिंक की प्रसार गति से प्रसारित होता है। प्रसार गति लिंक के भौतिक माध्यम (यानी, फाइबर ऑप्टिक्स, ट्विस्टेड-पेयर कॉपर वायर, इत्यादि) पर निर्भर करती है और इसकी सीमा होती है

2□108 मीटर/सेकंड से 3□108 मीटर/सेकंड

जो प्रकाश की गति के बराबर या उससे थोड़ा कम है। प्रसार विलंब प्रकाश के बीच की दूरी है

प्रसार गति से विभाजित दो राउटर। यानी प्रसार विलंब d/s है, जहाँ d राउटर A और राउटर B के बीच की दूरी है और s लिंक की प्रसार गति है। एक बार जब पैकेट का अंतिम बिट नोड B में प्रसारित हो जाता है, तो यह और पैकेट के सभी पूर्ववर्ती बिट राउटर B में संग्रहीत हो जाते हैं। पूरी प्रक्रिया तब जारी रहती है जब राउटर B अब अग्रेषण करता है। वाइड-एरिया नेटवर्क में, प्रसार विलंब मिलीसेकंड के क्रम में होता है।

संचरण और प्रसार विलंब की तुलना

VideoNote

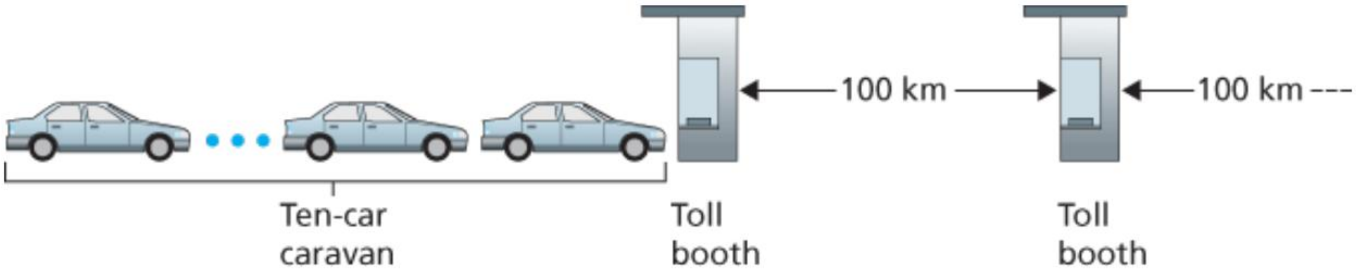
प्रसार विलंब और संचरण विलंब का अन्वेषण

कंप्यूटर नेटवर्किंग के क्षेत्र में नए लोगों को कभी-कभी ट्रांसमिशन देरी और प्रसार देरी के बीच अंतर को समझने में कठिनाई होती है। अंतर सूक्ष्म लेकिन महत्वपूर्ण है। ट्रांसमिशन देरी राउटर द्वारा पैकेट को पुश करने के लिए आवश्यक समय की मात्रा है; यह पैकेट की लंबाई और लिंक की ट्रांसमिशन दर का एक फंक्शन है, लेकिन दो राउटर के बीच की दूरी से इसका कोई लेना-देना नहीं है। दूसरी ओर, प्रसार देरी, वह समय है जो एक राउटर से दूसरे राउटर तक प्रसारित होने में लगता है; यह दो राउटर के बीच की दूरी का एक फंक्शन है, लेकिन पैकेट की लंबाई या लिंक की ट्रांसमिशन दर से इसका कोई लेना-देना नहीं है।

एक सादृश्य से संचरण और प्रसार विलंब की धारणा स्पष्ट हो सकती है। एक राजमार्ग पर विचार करें जिसमें

हर 100 किलोमीटर पर एक टोल बूथ, जैसा कि चित्र 1.17 में दिखाया गया है। आप राजमार्ग खंडों के बारे में सोच सकते हैं

टोलबूथ के बीच लिंक और टोलबूथ के बीच राउटर के रूप में। मान लीजिए कि कारें हाईवे पर 100 किमी/घंटा की गति से यात्रा करती हैं (यानी, फैलती हैं) (यानी, जब कोई कार टोलबूथ छोड़ती है, तो वह तुरंत 100 किमी/घंटा की गति पकड़ लेती है और टोलबूथ के बीच उस गति को बनाए रखती है)। मान लीजिए कि 10 कारें, एक कारवां के रूप में एक साथ यात्रा करते हुए, एक निश्चित क्रम में एक दूसरे का अनुसरण करती हैं। आप प्रत्येक कार को एक बिट और कारवां को एक पैकेट के रूप में सोच सकते हैं। यह भी मान लीजिए कि प्रत्येक



चित्र 1.17 कारवां सादृश्य

टोलबूथ एक कार को 12 सेकंड में एक कार की दर से सेवा प्रदान करता है (यानी, संचारित करता है), और यह कि रात का समय हो चुका है, इसलिए कारवां की कारें राजमार्ग पर एकमात्र कारें हैं। अंत में, मान लीजिए कि जब भी पहली कार

जब कारवां टोल बूथ पर पहुंचता है, तो वह प्रवेश द्वार पर तब तक इंतजार करता है जब तक कि अन्य नौ गाड़ियां नहीं पहुंच जातीं और

इसके पीछे कतार में खड़े हो जाएं। (इस प्रकार पूरे कारवां को आगे बढ़ने से पहले टोलबूथ पर संग्रहीत किया जाना चाहिए।) टोलबूथ को पूरे कारवां को राजमार्ग पर धकेलने के लिए आवश्यक समय (10 कारें) / (5 कारें / मिनट) = 2 मिनट है। यह समय राउटर में ट्रांसमिशन देरी के अनुरूप है। एक टोलबूथ के निकास से अगले टोलबूथ तक यात्रा करने के लिए कार के लिए आवश्यक समय 100 किमी / (100 किमी / घंटा) = 1 घंटा है। यह समय प्रसार विलंब के अनुरूप है। इसलिए, से समय

जब कारवां को एक टोल बूथ के सामने रखा जाता है तब तक जब तक कारवां अगले टोल बूथ के सामने न रखा जाए

संचरण विलंब और प्रसार विलंब का योग है - इस उदाहरण में, 62 मिनट।

आइए इस सादृश्य को थोड़ा और समझें। क्या होगा यदि कारवां के लिए टोलबूथ सेवा का समय कार द्वारा टोलबूथ के बीच यात्रा करने में लगने वाले समय से अधिक हो? उदाहरण के लिए, मान लीजिए कि अब कारें 1,000 किमी/घंटा की दर से यात्रा करती हैं और टोलबूथ एक मिनट में एक कार की दर से कारों की सेवा करता है। तो दो टोलबूथ के बीच यात्रा में देरी 6 मिनट है और कारवां की सेवा करने में 10 मिनट का समय लगता है।

इस मामले में, कारवां की पहली कुछ कारें अंतिम कारों से पहले दूसरे टोल बूथ पर पहुंच जाएंगी।

कारवां पहले टोलबूथ को छोड़ देता है। यह स्थिति पैकेट-स्विच्ड नेटवर्क में भी उत्पन्न होती है - पैकेट में पहले बिट्स राउटर पर पहुंच सकते हैं जबकि पैकेट में शेष बिट्स में से कई अभी भी पिछले राउटर द्वारा प्रेषित होने की प्रतीक्षा कर रहे हैं।

यदि एक चित्र हज़ारों शब्द बोलता है, तो एक एनीमेशन को लाखों शब्द बोलना चाहिए। इस पाठ्यपुस्तक की वेबसाइट एक इंटरैक्टिव जावा एप्लेट प्रदान करती है जो ट्रांसमिशन देरी को अच्छी तरह से चित्रित और विपरीत करती है

और प्रसार विलंब। पाठक को उस एप्लेट पर जाने के लिए अत्यधिक प्रोत्साहित किया जाता है। [स्मिथ 2009] प्रसार, कतार और संचरण विलंब की एक बहुत ही पठनीय चर्चा भी प्रदान करता है।

यदि हम ddd proc कतार ट्रांस , , , और d प्रसंस्करण, कतार, संचरण और प्रसार को दर्शाता है

देरी, तो कुल नोडल देरी द्वारा दिया जाता है

dnodal=dproc+dqueue+dtrans+dprop

इन विलंब घटकों का योगदान काफी भिन्न हो सकता है। उदाहरण के लिए, **एक ही विश्वविद्यालय पर दो राउटर को जोड़ने वाले लिंक के लिए** d प्रोप नगण्य (उदाहरण के लिए, कुछ माइक्रोसेकंड) हो सकता है

परिसर; हालाँकि, d एक भूस्थिर नेटवर्क द्वारा परस्पर जुड़े दो राउटरों के लिए सैकड़ों मिलीसेकंड है

सैटेलाइट लिंक, और d में प्रमुख शब्द हो सकता है । इसी तरह, d नगण्य से लेकर महत्वपूर्ण तक हो सकता है। 10 एमबीपीएस और उससे अधिक की ट्रांसमिशन दरों के लिए इसका योगदान आम तौर पर नगण्य है (उदाहरण के लिए, LAN के लिए); हालाँकि, यह भेजे जाने वाले बड़े इंटरनेट पैकेट के लिए सैकड़ों मिलीसेकंड हो सकता है

कम गति डायल-अप मॉडेम लिंक। प्रसंस्करण देरी, डी प्रोक. अक्सर नगण्य होता है; हालाँकि, यह दृढ़ता से राउटर के अधिकतम थ्रूपुट को प्रभावित करता है, जो कि वह अधिकतम दर है जिस पर राउटर पैकेट अग्रेषित कर सकता है।

1.4.2 कतारबद्ध विलंब और पैकेट हानि

नोडल विलंब का सबसे जटिल और दिलचस्प घटक कतार विलंब है, डी । वास्तव में, कतार विलंब कंप्यूटर नेटवर्किंग में इतना महत्वपूर्ण और दिलचस्प है कि हजारों पेपर और

इसके बारे में कई किताबें लिखी गई हैं [बर्टसेकस 1991; डेगल 1991; क्लेनरॉक 1975, क्लेनरॉक 1976; रॉस 1995]। हम यहाँ कतार में देरी के बारे में केवल एक उच्च-स्तरीय, सहज चर्चा दे रहे हैं; अधिक जिज्ञासु पाठक कुछ पुस्तकों को ब्राउज़ करना चाह सकते हैं (या अंततः इस पर पीएचडी थीसिस भी लिख सकते हैं)

विषय!)। अन्य तीन देरी (अर्थात, **पैकेट से पैकेट तक** डी) के विपरीत। उदाहरण के लिए, प्रोक. डी ट्रांस. और d), कतार में देरी अलग-अलग हो सकती है यदि 10 पैकेट एक ही समय में खाली कतार में पहुंचते हैं, तो प्रेषित पहला पैकेट कोई कतार देरी नहीं झेलेगा, जबकि प्रेषित अंतिम पैकेट अपेक्षाकृत बड़ी कतार देरी से पीड़ित होगा (जबकि यह अन्य नौ पैकेट प्रेषित होने की प्रतीक्षा करता है)। इसलिए, कतार में देरी को चिह्नित करते समय, आमतौर पर सांख्यिकीय उपायों का उपयोग किया जाता है, जैसे औसत कतार में देरी, कतार में देरी का विचरण, और संभावना है कि कतार में देरी कुछ निर्दिष्ट मूल्य से अधिक है।

कतार में देरी कब बड़ी होती है और कब यह नगण्य होती है? इस सवाल का जवाब इस बात पर निर्भर करता है कि ट्रैफ़िक कतार में किस दर से आता है, लिंक की ट्रांसमिशन दर और आने वाले ट्रैफ़िक की प्रकृति, यानी ट्रैफ़िक समय-समय पर आता है या फटने के रूप में आता है। कुछ जानकारी प्राप्त करने के लिए

यहाँ, a **को** उस औसत दर को दर्शाएँ जिस पर पैकेट कतार में पहुंचते हैं (a पैकेट/सेकंड की इकाई में है)।

याद रखें कि R संचरण दर है; अर्थात, यह वह दर (बिट्स/सेकंड में) है जिस पर बिट्स को कतार से बाहर धकेला जाता है। सरलता के लिए यह भी मान लें कि सभी पैकेट में L बिट्स होते हैं। तब बिट्स के कतार में पहुंचने की औसत दर La बिट्स/सेकंड होती है। अंत में, मान लें कि कतार बहुत बड़ी है, इसलिए यह अनिवार्य रूप से अनंत संख्या में बिट्स को रख सकती है। ट्रैफ़िक तीव्रता कहलाने वाला **अनुपात** La/R अक्सर कतार में लगने में होने वाली देरी की सीमा का अनुमान लगाने में महत्वपूर्ण भूमिका निभाता है। यदि La/R > 1 है, तो बिट्स के कतार में पहुंचने की औसत दर उस दर से अधिक होती है जिस पर बिट्स को कतार से प्रेषित किया जा सकता है। इसमें

दुर्भाग्यपूर्ण स्थिति में, कतार बिना सीमा के बढ़ती जाएगी और कतार में देरी निकट आ जाएगी

इसलिए, ट्रैफ़िक इंजीनियरिंग में एक सुनहरा नियम यह है: अपने सिस्टम को इस तरह डिज़ाइन करें कि ट्रैफ़िक की तीव्रता 1 से ज़्यादा न हो।

अब La/R □ 1 के मामले पर विचार करें । यहाँ, आने वाले ट्रैफ़िक की प्रकृति कतार में देरी को प्रभावित करती है। उदाहरण के लिए, यदि पैकेट समय-समय पर आते हैं - यानी, हर L/R सेकंड में एक पैकेट आता है - तो हर पैकेट खाली कतार में पहुंचेगा और कतार में कोई देरी नहीं होगी। दूसरी ओर, यदि पैकेट फटने के साथ-साथ लेकिन समय-समय पर आते हैं, तो कतार में देरी का एक महत्वपूर्ण औसत हो सकता है। उदाहरण के लिए, मान लीजिए

N पैकेट हर (L/R)N सेकंड में एक साथ आते हैं। फिर प्रेषित पहले पैकेट में कोई कतारबद्ध विलंब नहीं होता है; प्रेषित दूसरे पैकेट में L/R सेकंड की कतारबद्ध विलंब होता है; और अधिक सामान्यतः, **प्रेषित nवें (n−1) पैकेट** में L/R **सेकंड** की कतारबद्ध विलंब होता है । हम इस उदाहरण में औसत कतारबद्ध विलंब की गणना करने इसे आपके लिए एक अभ्यास के रूप में छोड़ते हैं। के लिए

ऊपर वर्णित आवधिक आगमन के दो उदाहरण थोड़े अकादमिक हैं। आम तौर पर, आगमन

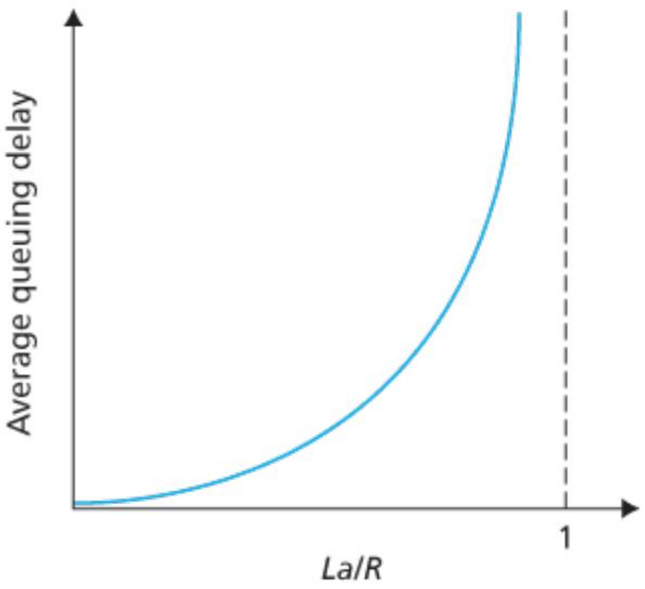
कतार में प्रक्रिया यादृच्छिक होती है; यानी, आगमन किसी पैटर्न का पालन नहीं करते हैं और पैकेट यादृच्छिक समय से एक दूसरे से दूरी पर होते हैं। इस अधिक यथार्थवादी मामले में, कतार में देरी के आंकड़ों को पूरी तरह से चिह्नित करने के लिए मात्रा La/R आमतौर पर पर्याप्त नहीं होती है। बहरहाल, कतार में देरी की सीमा की सहज समझ हासिल करने में यह उपयोगी है। विशेष रूप से, यदि ट्रैफ़िक तीव्रता शून्य के करीब है, तो पैकेट आगमन कम और दूर-दूर के होते हैं और यह असंभव है कि एक आने वाले पैकेट को कतार में कोई अन्य पैकेट मिल जाए। इसलिए, औसत कतार में देरी शून्य के करीब होगी। दूसरी ओर, जब ट्रैफ़िक तीव्रता 1 के करीब होती है, तो समय के अंतराल होंगे जब आगमन दर संचरण क्षमता (पैकेट आगमन दर में भिन्नता के कारण) से अधिक हो जाती है,

फिर भी, जैसे-जैसे यातायात की तीव्रता 1 के करीब पहुंचती है, कतार की औसत लंबाई और बड़ी होती जाती है।

यातायात तीव्रता पर औसत कतारबद्ध विलंब की गुणात्मक निर्भरता चित्र 1.18 में दर्शाई गई है।

चित्र 1.18 का एक महत्वपूर्ण पहलू यह है कि जैसे-जैसे ट्रैफ़िक की तीव्रता 1 के करीब पहुंचती है, कतार में लगने में औसत देरी तेज़ी से बढ़ती है। तीव्रता में एक छोटे प्रतिशत की वृद्धि से बहुत बड़ी देरी होगी।

देरी में प्रतिशत के हिसाब से वृद्धि। शायद आपने राजमार्ग पर इस घटना का अनुभव किया हो। यदि आप नियमित रूप से ऐसी सड़क पर गाड़ी चलाते हैं जो आमतौर पर भीड़भाड़ वाली होती है, तो यह तथ्य कि सड़क आमतौर पर



चित्र 1.18 यातायात की तीव्रता पर औसत कतारबद्ध विलंब की निर्भरता

भीड़भाड़ का अर्थ है कि इसकी यातायात तीव्रता 1 के करीब है। यदि किसी घटना के कारण सामान्य से थोड़ा भी अधिक यातायात होता है, तो आपको होने वाली देरी बहुत बड़ी हो सकती है।

कतार में देरी के बारे में वास्तव में अच्छी तरह से जानने के लिए, आपको एक बार फिर पाठ्यपुस्तक वेब साइट पर जाने के लिए प्रोत्साहित किया जाता है, जो कतार के लिए एक इंटरैक्टिव जावा एप्लेट प्रदान करता है। यदि आप पैकेट आगमन दर को इतना अधिक सेट करते हैं कि ट्रैफ़िक की तीव्रता 1 से अधिक हो जाती है, तो आप देखेंगे कि समय के साथ कतार धीरे-धीरे बढ़ती जाती है।

पैकेट हानि

ऊपर की हमारी चर्चाओं में, हमने मान लिया है कि कतार अनंत संख्या में पैकेट रखने में सक्षम है। वास्तव में, लिंक से पहले की कतार की क्षमता सीमित होती है, हालाँकि कतारबद्ध क्षमता राउटर के डिज़ाइन और लागत पर बहुत हद तक निर्भर करती है। चूँकि कतार की क्षमता सीमित होती है, इसलिए ट्रैफ़िक की तीव्रता 1 के करीब पहुँचने पर पैकेट विलंब वास्तव में अनंत तक नहीं पहुँचते। इसके बजाय, एक पैकेट पूरी कतार को खोजने के लिए आ सकता है।

ऐसे पैकेट को स्टोर करने के लिए कोई जगह न होने पर, राउटर उस पैकेट को छोड़ देगा ; यानी पैकेट खो जाएगा। कतार में यह ओवरफ़्लो फिर से जावा एप्लेट में कतार के लिए देखा जा सकता है जब ट्रैफ़िक की तीव्रता अधिक होती है
1 से अधिक.

एंड-सिस्टम के दृष्टिकोण से, पैकेट हानि एक पैकेट की तरह दिखाई देगी जो नेटवर्क कोर में प्रेषित की गई है लेकिन गंतव्य पर नेटवर्क से कभी नहीं निकलती है। ट्रैफ़िक की तीव्रता बढ़ने पर खोए हुए पैकेटों का अंश बढ़ता है। इसलिए, नोड पर प्रदर्शन को अक्सर न केवल देरी के संदर्भ में मापा जाता है, बल्कि पैकेट हानि की संभावना के संदर्भ में भी मापा जाता है। जैसा कि हम बाद के अध्यायों में चर्चा करेंगे, खोए हुए पैकेट को अंत-से-अंत आधार पर पुनः प्रेषित किया जा सकता है ताकि यह सुनिश्चित किया जा सके कि सभी डेटा अंततः स्रोत से गंतव्य तक स्थानांतरित हो जाएं।

1.4.3 अंत-से-अंत विलंब

अब तक हमारी चर्चा नोडल देरी पर केंद्रित रही है, यानी एक राउटर पर देरी। आइए अब स्रोत से गंतव्य तक कुल देरी पर विचार करें। इस अवधारणा को समझने के लिए, मान लें कि वहाँ

स्रोत होस्ट और गंतव्य होस्ट के बीच N−1 राउटर। चलिए फिलहाल यह भी मान लेते हैं कि ये हैं

नेटवर्क में भीड़भाड़ नहीं होनी चाहिए (ताकि कतार में लगने में देरी नगण्य हो), प्रत्येक पर प्रसंस्करण में देरी होनी चाहिए

रूटर और स्रोत होस्ट पर डी प्रोक है , प्रत्येक राउटर और स्रोत होस्ट से संचरण दर

R **बिट्स/सेकंड** है , और प्रत्येक लिंक पर प्रसार d **है** । नोडल विलंब एकत्रित होते हैं और अंत-से-अंत विलंब देते हैं,

$$d_{end-end}=N(d_{proc}+d_{trans}+d_{prop}) \tag{1.2}$$

एक बार फिर, जहाँ L पैकेट का आकार है। ध्यान दें कि $d_{trans}=L/R$, जहाँ समीकरण 1.2 समीकरण 1.1 का सामान्यीकरण है , **जिसमें प्रसंस्करण और प्रसार विलंब को ध्यान में नहीं रखा गया है।** नोड्स पर विषम विलंब के मामले में और प्रत्येक नोड पर औसत कतार विलंब की उपस्थिति के लिए समीकरण 1.2 को सामान्यीकृत करने का काम हम आप पर छोड़ते हैं।

ट्रेसरूट

VideoNote

नेटवर्क पथों की खोज करने और नेटवर्क विलंब को मापने के लिए ट्रेसरूट का उपयोग करना

कंप्यूटर नेटवर्क में एंड-टू-एंड देरी का अनुभव प्राप्त करने के लिए, हम ट्रेसरूट प्रोग्राम का उपयोग कर सकते हैं। ट्रेसरूट एक सरल प्रोग्राम है जो किसी भी इंटरनेट होस्ट में चलाया जा सकता है। जब उपयोगकर्ता गंतव्य होस्टनाम निर्दिष्ट करता है, तो स्रोत होस्ट में प्रोग्राम उस गंतव्य की ओर कई, विशेष पैकेट भेजता है। जैसे-जैसे ये पैकेट गंतव्य की ओर बढ़ते हैं, वे राउटर की एक श्रृंखला से गुजरते हैं। जब राउटर को इनमें से कोई विशेष पैकेट प्राप्त होता है, तो वह स्रोत को एक छोटा संदेश भेजता है जिसमें राउटर का नाम और पता होता है।

अधिक विशेष रूप से, मान लीजिए कि स्रोत और गंतव्य के बीच N−1 राउटर हैं। तब

स्रोत नेटवर्क में N **विशेष पैकेट** भेजेगा , **प्रत्येक पैकेट अंतिम स्रोत को संबोधित होगा।**

गंतव्य। इन N विशेष पैकेटों को 1 से N तक चिह्नित किया जाता है, जिसमें पहला पैकेट 1 से **चिह्नित होता है** और अंतिम पैकेट N से चिह्नित होता है। जब nth राउटर को n चिह्नित nth पैकेट **प्राप्त होता है, तो राउटर पैकेट को उसके गंतव्य की ओर अग्रेषित नहीं करता है, बल्कि स्रोत को वापस एक संदेश भेजता है। जब nth राउटर को nth पैकेट प्राप्त होता है ,**

तो राउटर पैकेट को उसके गंतव्य की ओर अग्रेषित नहीं करता है, बल्कि स्रोत को वापस एक संदेश भेजता है।

गंतव्य होस्ट को Nth पैकेट प्राप्त होता है, यह भी स्रोत को एक संदेश वापस देता है। स्रोत उस समय को रिकॉर्ड करता है जब वह पैकेट भेजता है और जब वह संबंधित पैकेट प्राप्त करता है

संदेश लौटाता है; यह संदेश लौटाने वाले राउटर (या गंतव्य होस्ट) का नाम और पता भी रिकॉर्ड करता है। इस तरीके से, स्रोत स्रोत से गंतव्य तक प्रवाहित होने वाले पैकेट द्वारा लिए गए मार्ग का पुनर्निर्माण कर सकता है, और स्रोत सभी मध्यवर्ती राउटरों के लिए राउंड-ट्रिप विलंब निर्धारित कर सकता है।

ट्रेसरूट वास्तव में वर्णित प्रयोग को तीन बार दोहराता है, इसलिए स्रोत वास्तव में 3 **भेजता है** •

गंतव्य तक N **पैकेट। RFC 1393 ट्रेसरूट का विस्तार से वर्णन करता है।**

यहाँ ट्रेसरूट प्रोग्राम के आउटपुट का एक उदाहरण दिया गया है, जहाँ रूट का पता लगाया जा रहा था

स्रोत होस्ट gaia.cs.umass.edu (मैसाचुसेट्स विश्वविद्यालय में) होस्ट cis.poly.edu **पर** (ब्रुकलिन में पॉलिटेक्निक यूनिवर्सिटी में)। आउटपुट में छह कॉलम हैं: पहला कॉलम **ऊपर वर्णित** n मान है, यानी रूट के साथ राउटर की संख्या; दूसरा कॉलम राउटर का नाम है; तीसरा कॉलम राउटर का पता है (xxx.xxx.xxx.xxx के रूप में); अंतिम तीन कॉलम तीन प्रयोगों के लिए राउंड-ट्रिप विलंब हैं। यदि स्रोत को किसी दिए गए राउटर से तीन से कम संदेश प्राप्त होते हैं (नेटवर्क में पैकेट हानि के कारण), तो ट्रेसरूट राउटर नंबर के ठीक बाद एक तारांकन चिह्न लगाता है और उस राउटर के लिए तीन से कम राउंड-ट्रिप समय की रिपोर्ट करता है।

```
1 सीएस-जीडब्ल्यू (128.119.240.254) 1.009 एमएस 0.899 एमएस 0.993 एमएस
2 128.119.3.154 (128.119.3.154) 0.931 एमएस 0.441 एमएस 0.651 एमएस
3 -border4-rt-gi-1-3.gw.umass.edu (128.119.2.194) 1.032 मि.से. 0.484 मि.से.
0.451 एमएस
4 -acr1-ge-2-1-0.Boston.cw.net (208.172.51.129) 10.006 एमएस 8.150 एमएस 8.460
एमएस
5 -agr4-loopback.NewYork.cw.net (206.24.194.104) 12.272 एमएस 14.344 एमएस
13.267 एमएस
6 -acr2-loopback.NewYork.cw.net (206.24.194.62) 13.225 एमएस 12.292 एमएस
12.148 एमएस
7 -pos10-2.core2.NewYork1.Level3.net (209.244.160.133) 12.218 एमएस 11.823
एमएस 11.793 एमएस
8 -gige9-1-52.hsipaccess1.NewYork1.Level3.net (64.159.17.39) 13.081 एमएस
11.556 मि.से. 13.297 मि.से.
9 -p0-0.polyu.bbnplanet.net (4.25.109.122) 12.716 मि.से. 13.052 मि.से. 12.786 मि.से.
10 cis.poly.edu (128.238.32.126) 14.080 मि.से. 13.035 मि.से. 12.802 मि.से.
```

ऊपर दिए गए ट्रेस में स्रोत और गंतव्य के बीच नौ राउटर हैं। इनमें से ज़्यादातर राउटर

सभी का एक नाम है, और उन सभी का पता है। उदाहरण के लिए, राउटर 3 का नाम border4-rt-gi- **है**

1-3.gw.umass.edu और इसका पता 128.119.2.194 **है** । इसी राउटर के लिए दिए गए डेटा को देखते हुए, हम देखते हैं कि तीन परीक्षणों में से पहले में स्रोत और राउटर के बीच राउंड-ट्रिप देरी 1.03 एमएसईसी थी। बाद के दो परीक्षणों के लिए राउंड-ट्रिप देरी 0.48 और 0.45 एमएसईसी थी। ये

राउंड-ट्रिप विलंब में अभी चर्चा की गई सभी देरी शामिल हैं, जिसमें ट्रांसमिशन देरी, प्रसार शामिल है
देरी, राउटर प्रोसेसिंग देरी और कतार में देरी। क्योंकि कतार में देरी समय के साथ बदलती रहती है,
राउटर n को भेजे गए पैकेट n की राउंड-ट्रिप देरी कभी-कभी राउटर n+1 को भेजे गए पैकेट n+1 की राउंड-ट्रिप देरी से ज़्यादा हो सकती है । वास्तव में, हम उपरोक्त उदाहरण में इस घटना को देखते हैं: राउटर 6 की देरी राउटर 7 की देरी से ज़्यादा है!

क्या आप खुद Traceroute आज़माना चाहते हैं? हम आपको http://www.traceroute.org पर जाने की सलाह देते हैं । जो
रूट ट्रेसिंग के लिए स्रोतों की एक विस्तृत सूची के लिए एक वेब इंटरफेस प्रदान करता है।
आप एक स्रोत चुनते हैं और किसी भी गंतव्य के लिए होस्टनाम प्रदान करते हैं। फिर ट्रेसरूट प्रोग्राम सारा काम करता है। ऐसे कई मुफ़्त सॉफ़्टवेयर प्रोग्राम हैं जो ग्राफ़िकल इंटरफ़ेस प्रदान करते हैं
ट्रेसरूट; हमारे पसंदीदा में से एक पिंगप्लॉटर [पिंगप्लॉटर 2016] है।

सिस्टम, एप्लिकेशन और अन्य विलंब समाप्त करें

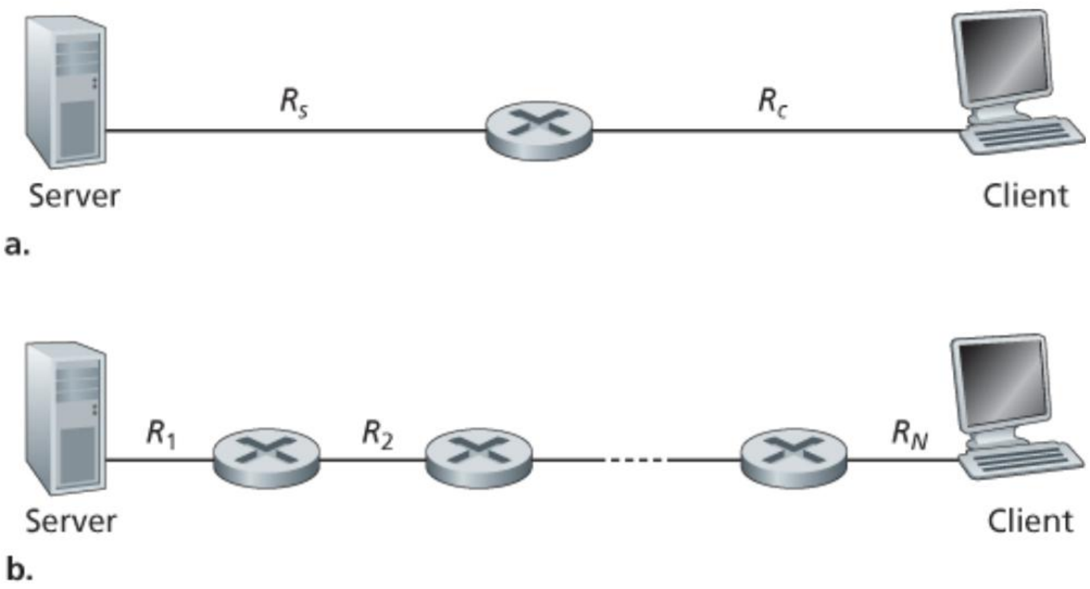
प्रसंस्करण, संचरण और प्रसार में देरी के अलावा, अंतिम सिस्टम में अतिरिक्त महत्वपूर्ण देरी हो सकती है। उदाहरण के लिए, एक अंतिम सिस्टम एक पैकेट को साझा नेटवर्क में संचारित करना चाहता है।
माध्यम (जैसे, वाईफाई या केबल मॉडेम परिदृश्य में) अन्य अंत प्रणालियों के साथ माध्यम को साझा करने के लिए अपने प्रोटोकॉल के भाग के रूप में जानबूझकर इसके प्रसारण में देरी कर सकता है; हम इस तरह के प्रोटोकॉल पर विस्तार से विचार करेंगे
अध्याय 6. एक अन्य महत्वपूर्ण विलंब मीडिया पैकेटाइजेशन विलंब है, जो वॉयस-ओवर-आईपी (वीओआईपी) अनुप्रयोगों में मौजूद है। वीओआईपी में, भेजने वाले पक्ष को पैकेट को इंटरनेट पर भेजने से पहले उसे एनकोडेड डिजिटाइज्ड स्पीच से भरना चाहिए। पैकेट भरने में लगने वाला यह समय - जिसे पैकेटाइजेशन विलंब कहा जाता है - महत्वपूर्ण हो सकता है और वीओआईपी कॉल की उपयोगकर्ता-अनुभूत गुणवत्ता को प्रभावित कर सकता है। इस मुद्दे को इस अध्याय के अंत में होमवर्क समस्या में आगे खोजा जाएगा।

1.4.4 कंप्यूटर नेटवर्क में थ्रूपुट

देरी और पैकेट हानि के अलावा, कंप्यूटर नेटवर्क में एक और महत्वपूर्ण प्रदर्शन माप एंड-टू-एंड थ्रूपुट है। थ्रूपुट को परिभाषित करने के लिए, कंप्यूटर नेटवर्क पर होस्ट ए से होस्ट बी तक एक बड़ी फ़ाइल स्थानांतरित करने पर विचार करें। यह स्थानांतरण, उदाहरण के लिए, एक पी2पी फ़ाइल शेयरिंग सिस्टम में एक पीयर से दूसरे पीयर तक एक बड़ी वीडियो क्लिप हो सकता है। किसी भी समय तात्कालिक थ्रूपुट वह दर (बिट्स/सेकंड में) है जिस पर होस्ट बी फ़ाइल प्राप्त कर रहा है। (कई एप्लिकेशन, जिनमें कई पी2पी फ़ाइल शेयरिंग सिस्टम शामिल हैं, उपयोगकर्ता इंटरफ़ेस में डाउनलोड के दौरान तात्कालिक थ्रूपुट प्रदर्शित करते हैं - शायद आपने इसे देखा हो

पहले!) यदि फ़ाइल में F बिट्स हैं और होस्ट B को सभी F बिट्स प्राप्त करने के लिए स्थानांतरण में T सेकंड लगते हैं, तो फ़ाइल स्थानांतरण का औसत थ्रूपुट F/T बिट्स/सेकंड है। कुछ अनुप्रयोगों के लिए, जैसे कि इंटरनेट टेलीफोनी, कम विलंब और कुछ सीमा से ऊपर लगातार तात्कालिक थ्रूपुट होना वांछनीय है (उदाहरण के लिए, कुछ इंटरनेट टेलीफोनी अनुप्रयोगों के लिए 24 kbps से अधिक और कुछ वास्तविक समय वीडियो अनुप्रयोगों के लिए 256 kbps से अधिक)। अन्य अनुप्रयोगों के लिए, जिनमें फ़ाइल स्थानांतरण शामिल हैं, विलंब महत्वपूर्ण नहीं है, लेकिन उच्चतम संभव थ्रूपुट होना वांछनीय है।

थ्रूपुट की महत्वपूर्ण अवधारणा के बारे में और अधिक जानकारी प्राप्त करने के लिए, आइए कुछ उदाहरणों पर विचार करें। चित्र 1.19(a) दो अंत प्रणालियों, एक सर्वर और एक क्लाइंट को दर्शाता है, जो दो संचार लिंक और एक द्वारा जुड़े हुए हैं राउटर। सर्वर से क्लाइंट तक फ़ाइल ट्रांसफ़र के लिए थ्रूपुट पर विचार करें। मान लें कि R फ़ाइल ट्रांसफ़र की दर को दर्शाता है सर्वर और राउटर के बीच लिंक; और R राउटर और सर्वर के बीच लिंक की दर को दर्शाता है। क्लाइंट। मान लीजिए कि पूरे नेटवर्क में केवल सर्वर से क्लाइंट तक ही बिट्स भेजे जा रहे हैं। अब हम पूछते हैं, इस आदर्श परिदृश्य में, सर्वर-से-क्लाइंट थ्रूपुट क्या है? इस प्रश्न का उत्तर देने के लिए, हम बिट्स को द्रव और संचार लिंक को पाइप के रूप में सोच सकते हैं। स्पष्ट रूप से, सर्वर बिट्स को पंप नहीं कर सकता **इसकी लिंक** R bps से अधिक तेज गति से अग्रेषित होती है ; और राउटर R bps से अधिक तेज गति से बिट्स को अग्रेषित नहीं कर सकता है। यदि रुपये<आरसी, तब सर्वर द्वारा पंप किए गए बिट्स राउटर के माध्यम से सीधे "प्रवाहित" होंगे और क्लाइंट तक पहुंचेंगे आर **बीपीएस** की दर , आर **बीपीएस** का थ्रूपुट दे रही है । अगर, दूसरी तरफ, तो राउटर नहीं होगा आरसी<रु, **बिट्स को जितनी जल्दी प्राप्त होता है उतनी ही जल्दी उन्हें अग्रेषित करने में सक्षम होता है। इस मामले में, बिट्स राउटर को केवल** R दर पर ही छोड़ेंगे , R का एंड-टू-एंड थ्रूपुट देना । **(यह भी ध्यान दें कि यदि बिट्स** R दर पर राउटर पर आना जारी रखते हैं , **और राउटर को** R पर छोड़ना जारी रखें , राउटर पर बिट्स का बैकलॉग प्रतीक्षा कर रहा है



चित्र 1.19 सर्वर से क्लाइंट तक फ़ाइल स्थानांतरण के लिए थ्रूपुट

क्लाइंट तक संचारण के लिए लागत बढ़ती ही जाएगी - एक सबसे अवांछनीय स्थिति!) इस प्रकार, इस सरल दो-लिंक नेटवर्क, थ्रूपुट न्यूनतम {R है , R }, अर्थात यह अड़चन लिंक की संचरण दर है।

थ्रूपुट निर्धारित करने के बाद, अब हम F **की एक बड़ी फ़ाइल को स्थानांतरित करने में लगने वाले समय का अनुमान लगा सकते हैं** सर्वर से क्लाइंट तक बिट्स को F/min{R , R } के रूप में भेजें। एक विशिष्ट उदाहरण के लिए, मान लीजिए कि आप MP3 डाउनलोड कर रहे हैं मिलियन बिट्स की फ़ाइल के लिए, सर्वर की संचरण दर F=32 Rs=2 है एमबीपीएस, और आपके पास एक एक्सेस लिंक है एमबीपीएस की। फ़ाइल को स्थानांतरित करने के लिए आवश्यक समय 32 सेकंड है। बेशक, इन अभिव्यक्तियों के लिए थ्रूपुट और ट्रांसफर समय केवल अनुमान हैं, क्योंकि वे स्टोर-एंड-फॉरवर्ड और प्रसंस्करण में देरी के साथ-साथ प्रोटोकॉल संबंधी समस्याएं भी उत्पन्न हो सकती हैं।

चित्र 1.19(b) **अब** सर्वर और क्लाइंट के बीच N **लिंक** वाला एक नेटवर्क दिखाता है , एन **लिंक** की संचरण दर दो-लिंक के लिए समान विश्लेषण लागू कर रही है आर1,आर2,..., आरएन. नेटवर्क, हम पाते हैं कि सर्वर से क्लाइंट तक फ़ाइल स्थानांतरण के लिए थ्रूपुट न्यूनतम {R1,R2,..., RN} है , कौन

एक बार फिर यह सर्वर और क्लाइंट के बीच पथ पर बाधा लिंक की संचरण दर है।

अब आज के इंटरनेट से प्रेरित एक और उदाहरण पर विचार करें। चित्र 1.20(a) दो अंतिम प्रणालियाँ दिखाता है, सर्वर और क्लाइंट, कंप्यूटर नेटवर्क से जुड़े हुए हैं। फ़ाइल ट्रांसफ़र के लिए थ्रूपुट पर विचार करें सर्वर क्लाइंट से जुड़ा हुआ है। सर्वर नेटवर्क से दर R **के एक्सेस लिंक के साथ जुड़ा हुआ है** और एस

क्लाइंट नेटवर्क से R **दर के एक्सेस लिंक के साथ जुड़ा हुआ है** । अब मान लीजिए कि नेटवर्क में सभी लिंक R दरों के एक्सेस लिंक के साथ जुड़े हुए हैं। संचार नेटवर्क के कोर में बहुत अधिक संचरण दर है, जो R और R **से भी बहुत अधिक है** । एस सी

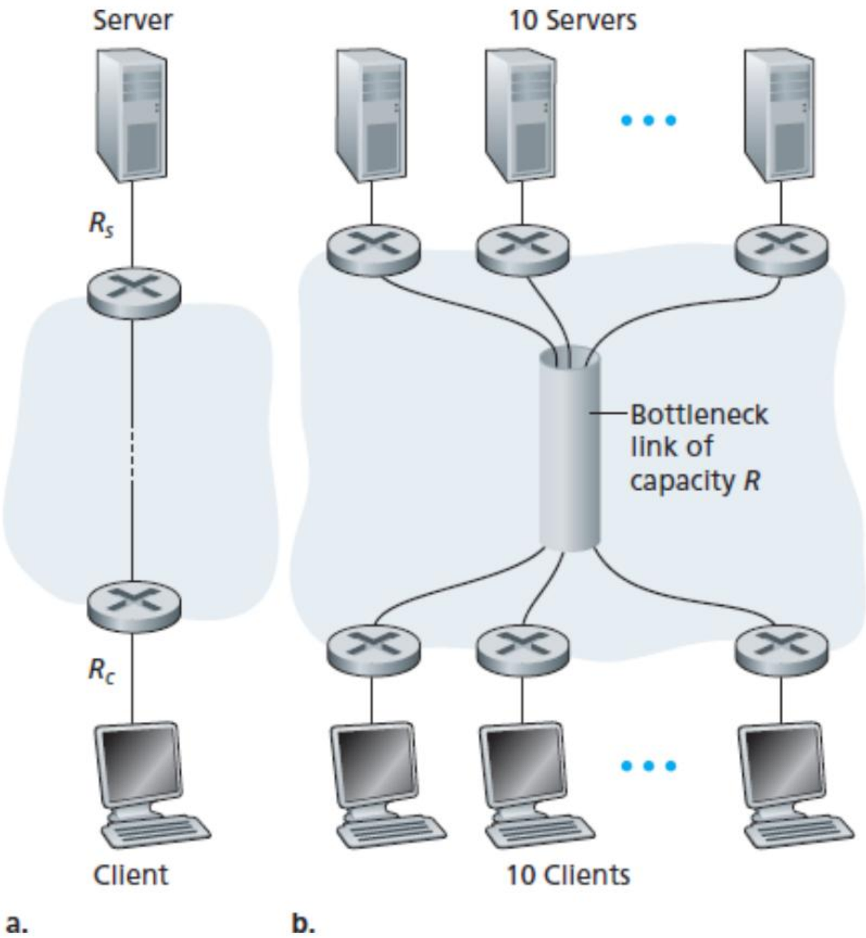
वास्तव में, आज इंटरनेट का मूल भाग उच्च गति वाले लिंकों से भरा हुआ है, जो बहुत कम अनुभव करते हैं। भीड़भाड़। यह भी मान लीजिए कि पूरे नेटवर्क में केवल सर्वर से सर्वर तक ही बिट्स भेजे जा रहे हैं क्लाइंट। क्योंकि इस उदाहरण में कंप्यूटर नेटवर्क का कोर एक चौड़ी पाइप की तरह है, इसलिए दर

कौन से बिट्स स्रोत से गंतव्य तक प्रवाहित हो सकते हैं, यह फिर से R और R का न्यूनतम है एस सी. यानी थ्रूपुट =

min{R , R }. इसलिए, आज के इंटरनेट में थ्रूपुट के लिए बाधक कारक आम तौर पर पहुँच है नेटवर्क।

अंतिम उदाहरण के लिए, चित्र 1.20(b) **पर विचार करें** जिसमें 10 सर्वर और 10 क्लाइंट एक दूसरे से जुड़े हुए हैं। कंप्यूटर नेटवर्क का मूल भाग। इस उदाहरण में, 10 डाउनलोड एक साथ हो रहे हैं, इसमें 10 क्लाइंट-सर्वर जोड़े शामिल हैं। मान लीजिए कि ये 10 डाउनलोड ही नेटवर्क में एकमात्र ट्रैफ़िक हैं वर्तमान समय। जैसा कि चित्र में दिखाया गया है, कोर में एक लिंक है जो सभी 10 डाउनलोड द्वारा पार किया जाता है। इस लिंक R की संचरण दर के लिए R को निरूपित करें । मान लीजिए कि सभी सर्वर एक्सेस लिंक में समान दर आर एस. सभी क्लाइंट एक्सेस लिंक की दर समान है R सी, और सभी लिंक की संचरण दर कोर - दर R की एक सामान्य कड़ी को छोड़कर - R से बहुत बड़े हैं एस. आरसी, और आर. अब हम पूछते हैं, क्या डाउनलोड के थ्रूपुट क्या हैं? स्पष्ट रूप से, यदि सामान्य लिंक की दर, R, बड़ी है - मान लीजिए कि R और R दोनों से सौ गुना बड़ा है - तो प्रत्येक डाउनलोड के लिए थ्रूपुट एक बार फिर होगा min{R , R }. **लेकिन क्या होगा यदि सामान्य लिंक की दर** R और R के समान क्रम की हो ? क्या होगा एस सी

इस मामले में थ्रूपुट क्या होगा? आइए एक विशिष्ट उदाहरण पर नज़र डालें। मान लीजिए Rs=2 एमबीपीएस, Rc=1 एमबीपीएस, आर=5 एमबीपीएस, और



चित्र 1.20 अंत-से-अंत थ्रूपुट: (ए) क्लाइंट सर्वर से फ़ाइल डाउनलोड करता है; (बी) 10 क्लाइंट 10 सर्वर से डाउनलोड करते हैं

कॉमन लिंक अपनी ट्रांसमिशन दर को 10 डाउनलोड के बीच बराबर बांटता है। फिर प्रत्येक डाउनलोड के लिए बाधा अब एक्सेस नेटवर्क में नहीं है, बल्कि अब कोर में साझा लिंक है, जो प्रत्येक डाउनलोड को केवल 500 केबीपीएस थ्रूपुट प्रदान करता है। इस प्रकार प्रत्येक डाउनलोड के लिए एंड-टू-एंड थ्रूपुट अब 500 केबीपीएस तक कम हो गया है।

चित्र 1.19 और चित्र 1.20(ए) में दिए गए उदाहरण दर्शाते हैं कि थ्रूपुट उन लिंक की ट्रांसमिशन दरों पर निर्भर करता है, जिन पर डेटा प्रवाहित होता है। हमने देखा कि जब कोई अन्य बीच का ट्रैफ़िक नहीं होता है, तो थ्रूपुट को केवल बीच के रास्ते पर न्यूनतम ट्रांसमिशन दर के रूप में अनुमानित किया जा सकता है।

स्रोत और गंतव्य। चित्र 1.20(बी) में उदाहरण से पता चलता है कि अधिक सामान्य रूप से थ्रूपुट न केवल पथ के साथ लिंक की संचरण दरों पर निर्भर करता है, बल्कि बीच के ट्रैफ़िक पर भी निर्भर करता है।

विशेष रूप से, उच्च संचरण दर वाला लिंक फिर भी फ़ाइल स्थानांतरण के लिए बाधा लिंक हो सकता है यदि कई अन्य डेटा प्रवाह भी उस लिंक से गुजर रहे हों। हम होमवर्क समस्याओं और बाद के अध्यायों में कंप्यूटर नेटवर्क में थ्रूपुट की अधिक बारीकी से जांच करेंगे।

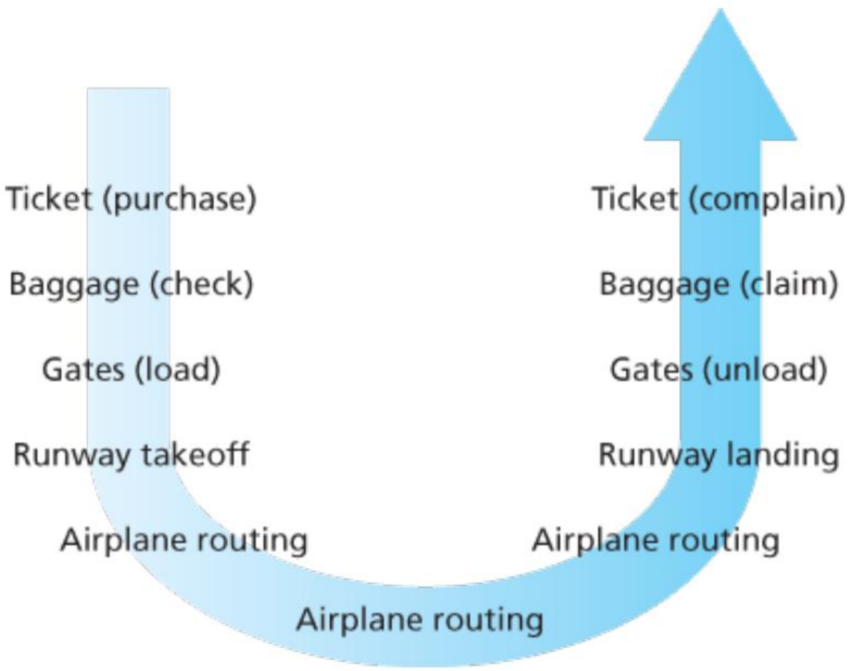
1.5 प्रोटोकॉल परतें और उनके सेवा मॉडल

अब तक की हमारी चर्चा से यह स्पष्ट है कि इंटरनेट एक अत्यंत जटिल प्रणाली है। हमने देखा है कि इंटरनेट के कई भाग हैं: कई अनुप्रयोग और प्रोटोकॉल, विभिन्न प्रकार के अंतिम सिस्टम, पैकेट स्विच और विभिन्न प्रकार के लिंक-लेवल मीडिया। इस अत्यधिक जटिलता को देखते हुए, क्या नेटवर्क आर्किटेक्चर को व्यवस्थित करने या कम से कम नेटवर्क आर्किटेक्चर पर हमारी चर्चा की कोई उम्मीद है? सौभाग्य से, दोनों प्रश्नों का उत्तर हाँ है।

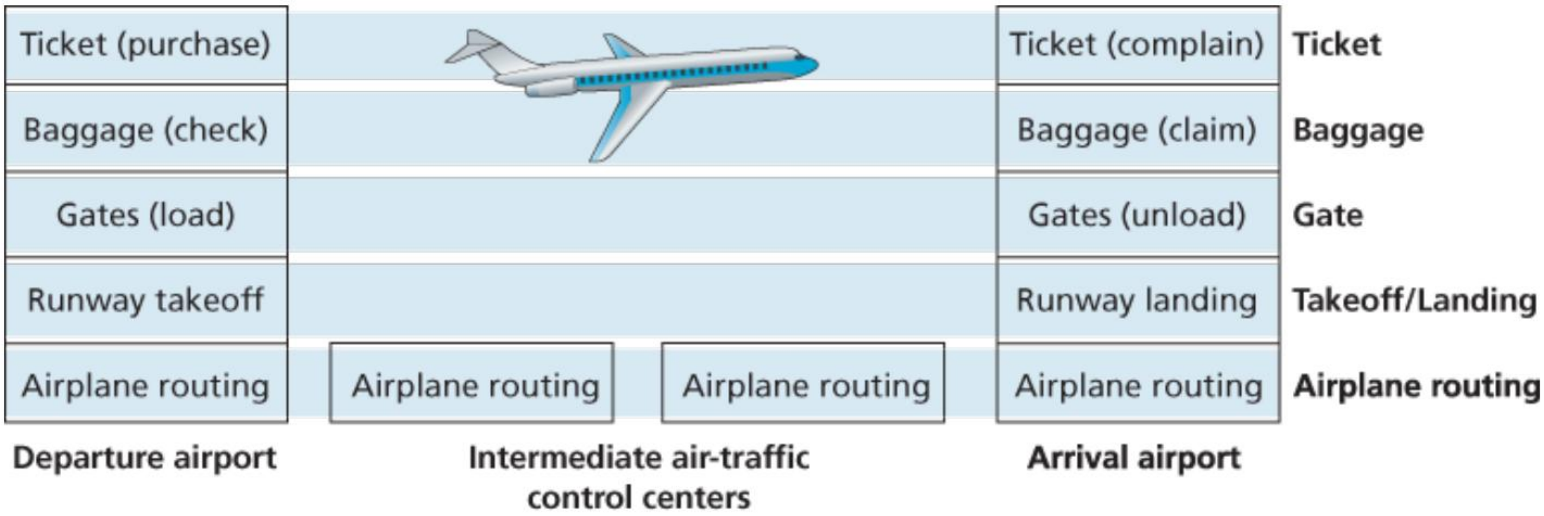
1.5.1 स्तरित वास्तुकला

इंटरनेट वास्तुकला पर अपने विचारों को व्यवस्थित करने का प्रयास करने से पहले, आइए एक मानवीय सादृश्य की तलाश करें।
दरअसल, हम अपने रोज़मर्रा के जीवन में हर समय जटिल प्रणालियों से निपटते हैं। कल्पना करें कि अगर कोई आपसे एयरलाइन सिस्टम का वर्णन करने के लिए कहे। आप इस जटिल सिस्टम का वर्णन करने के लिए संरचना कैसे ढूँढेंगे जिसमें टिकटिंग एजेंट, बैगेज चेकर, गेट कर्मचारी, पायलट, हवाई जहाज, एयर ट्रैफ़िक कंट्रोल और हवाई जहाज़ों को रूट करने के लिए एक विश्वव्यापी सिस्टम शामिल है? इस सिस्टम का वर्णन करने का एक तरीका यह हो सकता है कि आप एयरलाइन पर उड़ान भरते समय अपने द्वारा की जाने वाली क्रियाओं (या आपके लिए दूसरे लोग) की श्रृंखला का वर्णन करें। आप अपना टिकट खरीदते हैं, अपने बैग चेक करते हैं, गेट पर जाते हैं, और अंततः विमान में चढ़ जाते हैं। विमान उड़ान भरता है और अपने गंतव्य के लिए रवाना हो जाता है। आपके विमान के उतरने के बाद, आप गेट पर उतरते हैं और अपना बैग लेते हैं। अगर यात्रा खराब थी, तो आप टिकट एजेंट से उड़ान के बारे में शिकायत करते हैं (अपने प्रयास के लिए कुछ नहीं पाते)। यह परिदृश्य

चित्र 1.21 में दर्शाया गया है ।



चित्र 1.21 हवाई जहाज़ से यात्रा करना: क्रियाएँ



चित्र 1.22 एयरलाइन कार्यक्षमता की क्षैतिज परत

पहले से ही, हम यहाँ कंप्यूटर नेटवर्किंग के साथ कुछ समानताएं देख सकते हैं: आपको एयरलाइन द्वारा स्रोत से गंतव्य तक भेजा जा रहा है; एक पैकेट को स्रोत होस्ट से गंतव्य होस्ट तक एयरलाइन द्वारा भेजा जाता है।

इंटरनेट। लेकिन यह वह सादृश्य नहीं है जिसकी हम तलाश कर रहे हैं। हम चित्र 1.21 में कुछ संरचना की तलाश कर रहे हैं।

चित्र 1.21 को देखते हुए , हम देखते हैं कि प्रत्येक छोर पर एक टिकटिंग फ़ंक्शन है; पहले से टिकट वाले यात्रियों के लिए एक बैगेज फ़ंक्शन भी है, और पहले से टिकट वाले और पहले से बैगेज चेक किए गए यात्रियों के लिए एक गेट फ़ंक्शन भी है। गेट से गुज़रने वाले यात्रियों (यानी, जो पहले से टिकट वाले, बैगेज चेक किए गए और गेट से गुज़रे हुए यात्री हैं) के लिए, टेकऑफ़ और लैंडिंग फ़ंक्शन है, और उड़ान के दौरान, एक एयरप्लेन-रूटिंग फ़ंक्शन है। इससे पता चलता है कि हम कार्यक्षमता को देख सकते हैं

चित्र 1.21 को क्षैतिज रूप में रखें , जैसा कि चित्र 1.22 में दिखाया गया है।

चित्र 1.22 ने एयरलाइन की कार्यक्षमता को परतों में विभाजित किया है, जिससे एक रूपरेखा मिलती है जिसमें हम एयरलाइन यात्रा पर चर्चा कर सकते हैं। ध्यान दें कि प्रत्येक परत, उसके नीचे की परतों के साथ मिलकर, कुछ कार्यान्वयन करती है

कार्यक्षमता, कुछ सेवा। टिकटिंग लेयर और नीचे, किसी व्यक्ति का एयरलाइन-काउंटर-टू-एयरलाइन-काउंटर स्थानांतरण पूरा किया जाता है। बैगेज लेयर और नीचे, किसी व्यक्ति और बैग का बैगेज-चेक-टू-बैगेज-क्लेम स्थानांतरण पूरा किया जाता है। ध्यान दें कि बैगेज लेयर यह सेवा केवल पहले से टिकट वाले व्यक्ति को ही प्रदान करती है। गेट लेयर पर, किसी व्यक्ति और बैग का प्रस्थान-गेट-टू-आगमन-गेट स्थानांतरण पूरा किया जाता है। टेकऑफ़/लैंडिंग लेयर पर, लोगों और उनके बैग का रनवे-टू-रनवे स्थानांतरण पूरा किया जाता है। प्रत्येक लेयर (1) उस लेयर के भीतर कुछ क्रियाएं करके (उदाहरण के लिए, गेट लेयर पर, किसी हवाई जहाज से लोगों को उतारना और चढ़ाना) और (2) अपने से सीधे नीचे की लेयर की सेवाओं का उपयोग करके (उदाहरण के लिए, गेट लेयर में, टेकऑफ़/लैंडिंग लेयर की रनवे-टू-रनवे पैसेंजर ट्रांसफर सेवा का उपयोग करके) अपनी सेवा प्रदान करती है।

स्तरित वास्तुकला हमें एक बड़ी और जटिल प्रणाली के सुपरिभाषित, विशिष्ट भाग पर चर्चा करने की अनुमति देती है।
यह सरलीकरण अपने आप में मॉड्यूलरिटी प्रदान करके काफी मूल्यवान है, जिससे परत द्वारा प्रदान की गई सेवा के कार्यान्वयन को बदलना बहुत आसान हो जाता है। जब तक परत अपने ऊपर की परत को वही सेवा प्रदान करती है, और अपने नीचे की परत से वही सेवाएँ उपयोग करती है, तब तक जब परत के कार्यान्वयन को बदला जाता है, तो सिस्टम का शेष भाग अपरिवर्तित रहता है। (ध्यान दें कि परत के कार्यान्वयन को बदलने से सिस्टम का शेष भाग अपरिवर्तित रहता है।

किसी सेवा का क्रियान्वयन स्वयं सेवा को बदलने से बहुत अलग है!) उदाहरण के लिए, यदि गेट फ़ंक्शन बदल दिए गए (उदाहरण के लिए, लोगों को ऊंचाई के हिसाब से विमान में चढ़ाना और उतारना), तो एयरलाइन सिस्टम का शेष भाग अपरिवर्तित रहेगा क्योंकि गेट लेयर अभी भी वही फ़ंक्शन प्रदान करता है (लोगों को चढ़ाना और उतारना); यह केवल परिवर्तन के बाद उस फ़ंक्शन को एक अलग तरीके से लागू करता है। बड़े और जटिल सिस्टम के लिए जिन्हें लगातार अपडेट किया जा रहा है, सिस्टम के अन्य घटकों को प्रभावित किए बिना किसी सेवा के कार्यान्वयन को बदलने की क्षमता लेयरिंग का एक और महत्वपूर्ण लाभ है।

प्रोटोकॉल लेयरिंग

लेकिन एयरलाइन्स के बारे में इतना ही काफी है। चलिए अब अपना ध्यान नेटवर्क प्रोटोकॉल पर केंद्रित करते हैं। नेटवर्क प्रोटोकॉल के डिज़ाइन को संरचना प्रदान करने के लिए, नेटवर्क डिज़ाइनर प्रोटोकॉल को व्यवस्थित करते हैं - और नेटवर्क हार्डवेयर और सॉफ़्टवेयर जो प्रोटोकॉल को लागू करते हैं - परतों में। प्रत्येक प्रोटोकॉल एक परत से संबंधित है, ठीक उसी तरह जैसे कि

चित्र 1.22 में एयरलाइन आर्किटेक्चर में प्रत्येक फ़ंक्शन एक परत से संबंधित था। हम फिर से उन सेवाओं **में रुचि रखते हैं** जो एक परत ऊपर की परत को प्रदान करती है - एक परत का तथाकथित सेवा मॉडल । हमारे एयरलाइन उदाहरण के मामले की तरह, प्रत्येक परत अपनी सेवा (1) उस परत के भीतर कुछ क्रियाएँ करके और (2) सीधे उसके नीचे की परत की सेवाओं का उपयोग करके प्रदान करती है। उदाहरण के लिए, प्रदान की गई सेवाएँ

लेयर n **द्वारा** नेटवर्क के एक किनारे से दूसरे किनारे तक संदेशों की विश्वसनीय डिलीवरी शामिल हो सकती है। इसे लेयर n−1 की अविश्वसनीय एज-टू-एज मैसेज डिलीवरी सेवा का उपयोग करके लागू किया जा सकता है , और

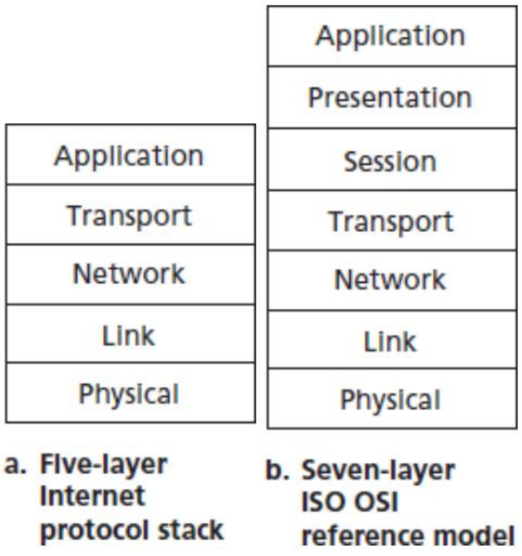
खोए हुए संदेशों का पता लगाने और पुनः प्रेषित करने के लिए परत एन कार्यक्षमता जोड़ना ।

प्रोटोकॉल परत को सॉफ्टवेयर, हार्डवेयर या दोनों के संयोजन में क्रियान्वित किया जा सकता है।
एप्लिकेशन-लेयर प्रोटोकॉल - जैसे HTTP और SMTP - लगभग हमेशा अंतिम सिस्टम में सॉफ़्टवेयर में कार्यान्वित किए जाते हैं; ट्रांसपोर्ट-लेयर प्रोटोकॉल भी ऐसे ही हैं। चूँकि भौतिक परत और डेटा लिंक परतें किसी विशिष्ट लिंक पर संचार को संभालने के लिए ज़िम्मेदार होती हैं, इसलिए उन्हें आमतौर पर किसी दिए गए लिंक से जुड़े नेटवर्क इंटरफ़ेस कार्ड (उदाहरण के लिए, ईथरनेट या WiFi इंटरफ़ेस कार्ड) में कार्यान्वित किया जाता है। नेटवर्क परत अक्सर हार्डवेयर और सॉफ़्टवेयर का मिश्रित कार्यान्वयन होता है। यह भी ध्यान दें कि जिस तरह से लेयर्ड एयरलाइन आर्किटेक्चर में फ़ंक्शन विभिन्न हवाई अड्डों और उड़ान नियंत्रण केंद्रों के बीच वितरित किए गए थे, उसी तरह से लेयर्ड एयरलाइन आर्किटेक्चर में भी कई तरह के काम किए गए थे।

सिस्टम बनाते हैं, इसलिए लेयर एन प्रोटोकॉल **भी एंड सिस्टम, पैकेट स्विच और नेटवर्क बनाने वाले अन्य घटकों के बीच** वितरित किया जाता है । यानी, इन नेटवर्क घटकों में से प्रत्येक में अक्सर लेयर एन प्रोटोकॉल का एक टुकड़ा होता है।

प्रोटोकॉल लेयरिंग के वैचारिक और संरचनात्मक लाभ हैं [RFC 3439]। जैसा कि हमने देखा है, लेयरिंग सिस्टम घटकों पर चर्चा करने का एक संरचित तरीका प्रदान करता है। मॉड्यूलरिटी सिस्टम घटकों को अपडेट करना आसान बनाती है। हालाँकि, हम उल्लेख करते हैं कि कुछ शोधकर्ता और नेटवर्किंग इंजीनियर ज़ोरदार तरीके से इस पर चर्चा करते हैं।

लेयरिंग के विपरीत [वेकमैन 1992]। लेयरिंग का एक संभावित दोष यह है कि एक परत निचली परत की कार्यक्षमता को दोहरा सकती है। उदाहरण के लिए, कई प्रोटोकॉल स्टैक त्रुटि पुनर्प्राप्ति प्रदान करते हैं



चित्र 1.23 इंटरनेट प्रोटोकॉल स्टैक (ए) और ओएसआई संदर्भ मॉडल (बी)

प्रति-लिंक आधार पर और अंत-से-अंत आधार पर। दूसरा संभावित दोष यह है कि एक परत पर कार्यक्षमता के लिए ऐसी जानकारी की आवश्यकता हो सकती है (उदाहरण के लिए, टाइमस्टैम्प मान) जो केवल दूसरी परत में मौजूद हो; यह परतों के पृथक्करण के लक्ष्य का उल्लंघन करता है।

जब एक साथ लिया जाता है, तो विभिन्न परतों के प्रोटोकॉल को प्रोटोकॉल स्टैक कहा जाता है। इंटरनेट प्रोटोकॉल स्टैक में पाँच परतें होती हैं: भौतिक, लिंक, नेटवर्क, ट्रांसपोर्ट और एप्लिकेशन परतें, जैसा कि नीचे दिखाया गया है।

चित्र 1.23(a) में दिखाया गया है । यदि आप विषय-सूची की जांच करते हैं, तो आप देखेंगे कि हमने इंटरनेट प्रोटोकॉल स्टैक की परतों का उपयोग करके इस पुस्तक को मोटे तौर पर व्यवस्थित किया है। हम ऊपर से नीचे की ओर दृष्टिकोण अपनाते हैं, पहले एप्लिकेशन परत को कवर करते हैं और फिर नीचे की ओर बढ़ते हैं।

अनुप्रयोग परत

एप्लीकेशन लेयर वह जगह है जहाँ नेटवर्क एप्लीकेशन और उनके एप्लीकेशन-लेयर प्रोटोकॉल रहते हैं। इंटरनेट की एप्लीकेशन लेयर में कई प्रोटोकॉल शामिल हैं, जैसे HTTP प्रोटोकॉल (जो वेब डॉक्यूमेंट रिक्वेस्ट और ट्रांसफर के लिए उपलब्ध कराता है), SMTP (जो ई-मेल मैसेज के ट्रांसफर के लिए उपलब्ध कराता है) और FTP (जो दो एंड सिस्टम के बीच फाइलों के ट्रांसफर के लिए उपलब्ध कराता है)। हम देखेंगे कि कुछ नेटवर्क

कार्य, जैसे कि www.ietf.org जैसी इंटरनेट एंड प्रणालियों के लिए मानव-अनुकूल नामों का अनुवाद 32-बिट नेटवर्क पते पर स्थानांतरण, एक विशिष्ट अनुप्रयोग-स्तर प्रोटोकॉल की सहायता से भी किया जाता है, अर्थात्,

डोमेन नाम प्रणाली (DNS) के बारे में अधिक जानकारी के लिए, अध्याय 2 में देखें कि अपने स्वयं के नए एप्लिकेशन-लेयर प्रोटोकॉल बनाना और उन्हें तैनात करना बहुत आसान है।

एप्लिकेशन-लेयर प्रोटोकॉल कई एंड सिस्टम पर वितरित किया जाता है, जिसमें एक एंड सिस्टम में एप्लिकेशन दूसरे एंड सिस्टम में एप्लिकेशन के साथ सूचना के पैकेट का आदान-प्रदान करने के लिए प्रोटोकॉल का उपयोग करता है। हम एप्लिकेशन लेयर पर सूचना के इस पैकेट को संदेश के रूप में संदर्भित करेंगे।

परिवहन परत

इंटरनेट का ट्रांसपोर्ट लेयर एप्लीकेशन-लेयर मैसेज को एप्लीकेशन एंडपॉइंट्स के बीच ट्रांसपोर्ट करता है। इंटरनेट में दो ट्रांसपोर्ट प्रोटोकॉल हैं, TCP और UDP, जिनमें से कोई भी एप्लीकेशन-लेयर मैसेज को ट्रांसपोर्ट कर सकता है। TCP अपने एप्लीकेशन को कनेक्शन-ओरिएंटेड सर्विस प्रदान करता है। इस सर्विस में एप्लीकेशन-लेयर मैसेज को गंतव्य तक पहुंचाने की गारंटी और फ्लो कंट्रोल (यानी, प्रेषक/रिसीवर स्पीड मैचिंग) शामिल है। TCP लंबे मैसेज को छोटे सेगमेंट में भी तोड़ता है और कंजेशन-कंट्रोल मैकेनिज्म प्रदान करता है, ताकि जब नेटवर्क कंजस्टेड हो तो सोर्स अपनी ट्रांसमिशन दर को कम कर सके। UDP प्रोटोकॉल अपने एप्लीकेशन को कनेक्शन रहित सर्विस प्रदान करता है। यह एक नो-फ्रिल्स सर्विस है जो कोई विश्वसनीयता, कोई फ्लो कंट्रोल और कोई कंजेशन कंट्रोल प्रदान नहीं करती है। इस पुस्तक में, हम ट्रांसपोर्ट-लेयर पैकेट को सेगमेंट के रूप में संदर्भित करेंगे।

नेटवर्क परत

इंटरनेट की नेटवर्क परत नेटवर्क-लेयर पैकेट को डेटाग्राम **के रूप में जाना जाता है** जो एक होस्ट से दूसरे होस्ट तक ले जाने के लिए जिम्मेदार है। एक स्रोत होस्ट में इंटरनेट ट्रांसपोर्ट-लेयर प्रोटोकॉल (TCP या UDP) एक होस्ट से दूसरे होस्ट तक जाता है।
ट्रांसपोर्ट लेयर सेगमेंट और नेटवर्क लेयर को गंतव्य पता, ठीक वैसे ही जैसे आप देते हैं
डाक सेवा गंतव्य पते के साथ एक पत्र। नेटवर्क परत तब सेवा प्रदान करती है
गंतव्य होस्ट में परिवहन परत तक सेगमेंट को पहुंचाना।

इंटरनेट की नेटवर्क परत में प्रसिद्ध आईपी प्रोटोकॉल शामिल है, जो डेटाग्राम में फ़ील्ड को परिभाषित करता है और साथ ही यह भी बताता है कि अंतिम सिस्टम और राउटर इन फ़ील्ड पर कैसे कार्य करते हैं। केवल एक आईपी प्रोटोकॉल है, और सभी
इंटरनेट घटकों में नेटवर्क परत होती है, उन्हें IP प्रोटोकॉल चलाना चाहिए। इंटरनेट की नेटवर्क परत
इसमें रूटिंग प्रोटोकॉल भी शामिल हैं जो डेटाग्राम के स्रोतों और नेटवर्क के बीच के रूट को निर्धारित करते हैं।
गंतव्य। इंटरनेट में कई रूटिंग प्रोटोकॉल हैं। जैसा कि हमने सेक्शन 1.3 में देखा, इंटरनेट नेटवर्कों का एक नेटवर्क है, और एक नेटवर्क के भीतर, नेटवर्क एडमिनिस्ट्रेटर कोई भी रूटिंग प्रोटोकॉल चला सकता है
वांछित। हालाँकि नेटवर्क परत में आईपी प्रोटोकॉल और कई रूटिंग प्रोटोकॉल दोनों शामिल हैं, यह
इसे अक्सर आईपी परत के रूप में संदर्भित किया जाता है, जो इस तथ्य को दर्शाता है कि आईपी वह गोंद है जो इंटरनेट को बांधता है
एक साथ।

लिंक परत

इंटरनेट की नेटवर्क परत स्रोत और गंतव्य के बीच राउटर की एक श्रृंखला के माध्यम से डेटाग्राम को रूट करती है। एक पैकेट को एक नोड (होस्ट या राउटर) से रूट में अगले नोड तक ले जाने के लिए, नेटवर्क परत लिंक परत की सेवाओं पर निर्भर करती है। विशेष रूप से, प्रत्येक नोड पर, नेटवर्क परत डेटाग्राम को लिंक परत तक पहुंचाती है, जो डेटाग्राम को रूट के साथ अगले नोड तक पहुंचाती है। इस अगले नोड पर, लिंक परत डेटाग्राम को नेटवर्क परत तक पहुंचाती है।

लिंक लेयर द्वारा प्रदान की जाने वाली सेवाएँ उस विशिष्ट लिंक-लेयर प्रोटोकॉल पर निर्भर करती हैं जिसे लिंक पर नियोजित किया जाता है। उदाहरण के लिए, कुछ लिंक-लेयर प्रोटोकॉल एक लिंक पर संचारित नोड से प्राप्त नोड तक विश्वसनीय डिलीवरी प्रदान करते हैं। ध्यान दें कि यह विश्वसनीय डिलीवरी सेवा TCP की विश्वसनीय डिलीवरी सेवा से अलग है, जो एक एंड सिस्टम से दूसरे एंड सिस्टम तक विश्वसनीय डिलीवरी प्रदान करती है। लिंक-लेयर के उदाहरण

प्रोटोकॉल में ईथरनेट, वाईफ़ाई और केबल एक्सेस नेटवर्क का DOCSIS प्रोटोकॉल शामिल है। चूंकि डेटाग्राम को आम तौर पर स्रोत से गंतव्य तक यात्रा करने के लिए कई लिंक पार करने की आवश्यकता होती है, इसलिए डेटाग्राम को उसके मार्ग के विभिन्न लिंक पर विभिन्न लिंक-लेयर प्रोटोकॉल द्वारा नियंत्रित किया जा सकता है। उदाहरण के लिए, एक डेटाग्राम को एक लिंक पर ईथरनेट द्वारा और अगले लिंक पर PPP द्वारा नियंत्रित किया जा सकता है। नेटवर्क लेयर को प्रत्येक अलग-अलग लिंक-लेयर प्रोटोकॉल से एक अलग सेवा प्राप्त होगी। इस पुस्तक में, हम लिंक-लेयर पैकेट को इस प्रकार संदर्भित करेंगे

फ्रेम.

भौतिक परत

जबकि लिंक लेयर का काम पूरे फ्रेम को एक नेटवर्क तत्व से दूसरे आसन्न नेटवर्क पर ले जाना है

तत्व, भौतिक परत का काम **फ्रेम के भीतर** अलग-अलग बिट्स को एक नोड से दूसरे नोड तक ले जाना है। इस परत में प्रोटोकॉल फिर से लिंक पर निर्भर होते हैं और आगे लिंक के वास्तविक ट्रांसमिशन माध्यम (उदाहरण के लिए, ट्विस्टेड-पेयर कॉपर वायर, सिंगल-मोड फाइबर ऑप्टिक्स) पर निर्भर करते हैं। उदाहरण के लिए, ईथरनेट में कई भौतिक-परत प्रोटोकॉल हैं: एक ट्विस्टेड-पेयर कॉपर वायर के लिए, दूसरा कोएक्सियल केबल के लिए, दूसरा फाइबर के लिए, और इसी तरह। प्रत्येक मामले में, एक बिट को लिंक पर अलग-अलग तरीके से ले जाया जाता है।

ओएसआई मॉडल

इंटरनेट प्रोटोकॉल स्टैक पर विस्तार से चर्चा करने के बाद, हमें यह उल्लेख करना चाहिए कि यह एकमात्र प्रोटोकॉल स्टैक नहीं है। विशेष रूप से, 1970 के दशक के उत्तरार्ध में, अंतर्राष्ट्रीय मानकीकरण संगठन (ISO) ने प्रस्ताव दिया था कि कंप्यूटर नेटवर्क को सात परतों के आसपास व्यवस्थित किया जाना चाहिए, जिन्हें ओपन सिस्टम कहा जाता है।

इंटरकनेक्शन (OSI) मॉडल [ISO 2016]। OSI मॉडल ने तब आकार लिया जब इंटरनेट प्रोटोकॉल बनने वाले प्रोटोकॉल अपनी प्रारंभिक अवस्था में थे, और विकास के तहत कई अलग-अलग प्रोटोकॉल सूट में से एक थे; वास्तव में, मूल OSI मॉडल के आविष्कारकों ने इसे बनाते समय शायद इंटरनेट को ध्यान में नहीं रखा था। फिर भी, 1970 के दशक के उत्तरार्ध में, कई प्रशिक्षण और विश्वविद्यालय पाठ्यक्रमों ने ISO अधिदेश को अपनाया और सात-परत मॉडल के आसपास पाठ्यक्रम आयोजित किए। नेटवर्किंग शिक्षा पर इसके शुरुआती प्रभाव के कारण, सात-परत मॉडल कुछ नेटवर्किंग पाठ्यपुस्तकों और प्रशिक्षण पाठ्यक्रमों में अभी भी मौजूद है।

चित्र 1.23(बी) में दिखाए गए ओएसआई संदर्भ मॉडल की सात परतें हैं: एप्लीकेशन लेयर, प्रेजेंटेशन लेयर, सेशन लेयर, ट्रांसपोर्ट लेयर, नेटवर्क लेयर, डेटा लिंक लेयर और फिजिकल लेयर। इनमें से पाँच लेयर की कार्यक्षमता मोटे तौर पर उनके समान नाम वाले इंटरनेट समकक्षों के समान ही है।

इस प्रकार, आइए OSI संदर्भ मॉडल में मौजूद दो अतिरिक्त परतों पर विचार करें- प्रस्तुति परत और सत्र परत। प्रस्तुति परत की भूमिका ऐसी सेवाएँ प्रदान करना है जो संचार अनुप्रयोगों को आदान-प्रदान किए गए डेटा के अर्थ की व्याख्या करने की अनुमति देती हैं। इन सेवाओं में डेटा संपीड़न और डेटा एन्क्रिप्शन (जो स्वयं-व्याख्यात्मक हैं) के साथ-साथ डेटा विवरण (जो अनुप्रयोगों को उस आंतरिक प्रारूप के बारे में चिंता करने से मुक्त करता है जिसमें डेटा का प्रतिनिधित्व/संग्रह किया जाता है - प्रारूप जो एक कंप्यूटर से दूसरे कंप्यूटर में भिन्न हो सकते हैं) शामिल हैं। सत्र परत डेटा एक्सचेंज के सीमांकन और सिंक्रनाइज़ेशन के लिए प्रदान करती है, जिसमें चेकपॉइंटिंग और रिकवरी योजना बनाने के साधन शामिल हैं।

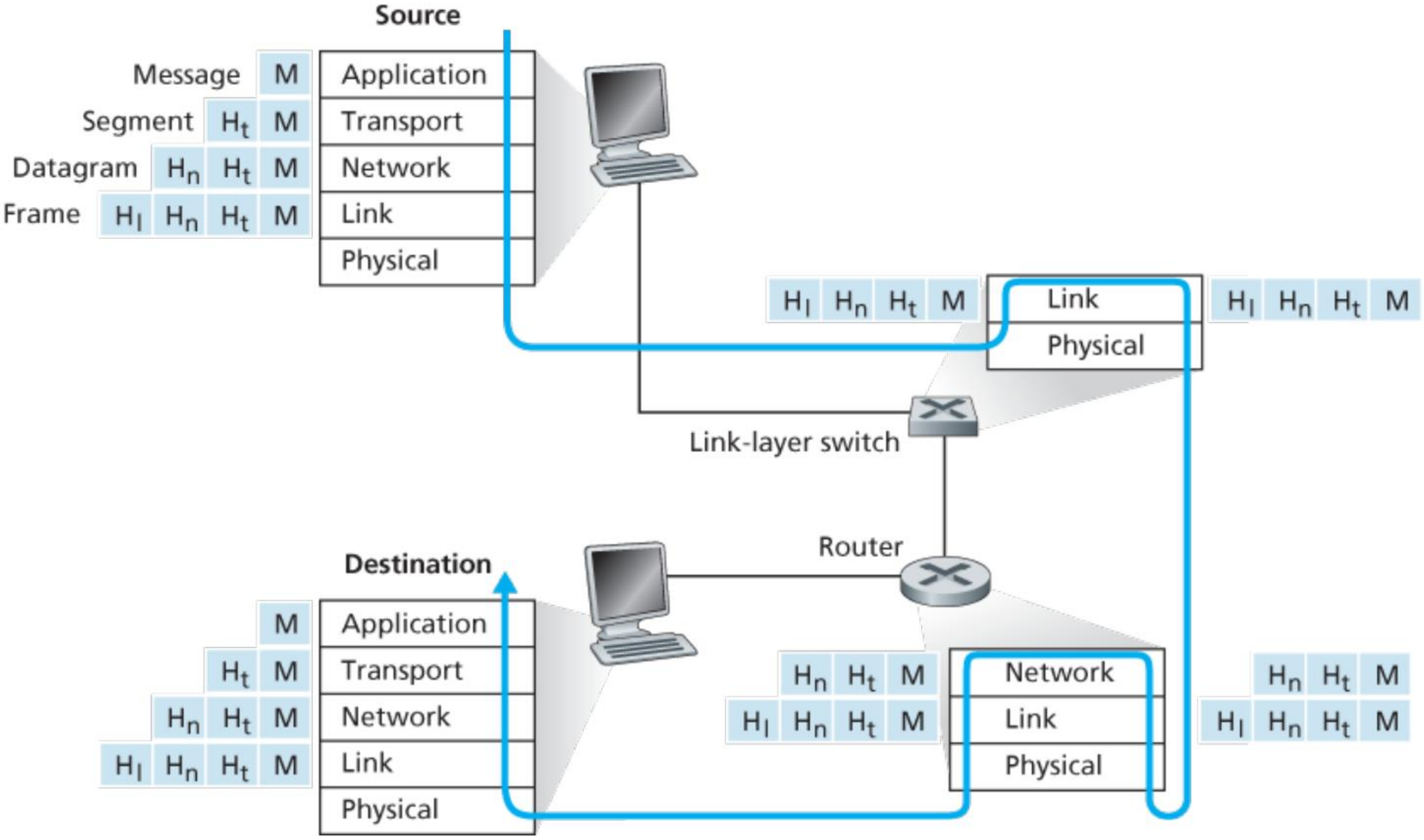
यह तथ्य कि इंटरनेट में OSI संदर्भ मॉडल में पाई जाने वाली दो परतों का अभाव है, कुछ दिलचस्प सवाल खड़े करता है: क्या इन परतों द्वारा प्रदान की जाने वाली सेवाएँ महत्वहीन हैं? क्या होगा यदि कोई एप्लिकेशन

इनमें से किसी एक सेवा की ज़रूरत है ? इन दोनों सवालों के लिए इंटरनेट का जवाब एक ही है - यह एप्लिकेशन डेवलपर पर निर्भर करता है। यह एप्लिकेशन डेवलपर पर निर्भर करता है कि वह तय करे कि कोई सेवा महत्वपूर्ण है या नहीं और क्या यह ज़रूरी है।

सेवा **महत्वपूर्ण** है , लेकिन उस कार्यक्षमता को अनुप्रयोग में शामिल करना अनुप्रयोग डेवलपर पर निर्भर है।

1.5.2 एनकैप्सुलेशन

चित्र 1.24 उस भौतिक पथ को दर्शाता है जिस पर डेटा प्रेषण अंत प्रणाली के प्रोटोकॉल स्टैक से नीचे, तथा मध्यवर्ती लिंक-लेयर स्विच के प्रोटोकॉल स्टैक से ऊपर और नीचे जाता है



चित्र 1.24 होस्ट, राउटर और लिंक-लेयर स्विच; प्रत्येक में परतों का एक अलग सेट होता है, जो उनकी कार्यक्षमता में अंतर को दर्शाता है

और राउटर, और फिर रिसीविंग एंड सिस्टम पर प्रोटोकॉल स्टैक को ऊपर ले जाते हैं। जैसा कि हम इस पुस्तक में आगे चर्चा करेंगे, राउटर और लिंक-लेयर स्विच दोनों पैकेट स्विच हैं। एंड सिस्टम की तरह, राउटर और लिंक-लेयर स्विच अपने नेटवर्किंग हार्डवेयर और सॉफ़्टवेयर को परतों में व्यवस्थित करते हैं। लेकिन राउटर और लिंक-लेयर

स्विच प्रोटोकॉल स्टैक में सभी **परतों को** लागू नहीं करते हैं ; **वे आम तौर पर केवल निचली परतों को लागू करते हैं। जैसा कि** चित्र 1.24 में दिखाया गया है, लिंक-लेयर स्विच परत 1 और 2 को लागू करते हैं; राउटर परत 1 से 3 को लागू करते हैं। इसका मतलब है, उदाहरण के लिए, कि इंटरनेट राउटर आईपी प्रोटोकॉल (एक परत 3 प्रोटोकॉल) को लागू करने में सक्षम हैं, जबकि लिंक-लेयर स्विच नहीं हैं। हम बाद में देखेंगे कि

जबकि लिंक-लेयर स्विच आईपी एड्रेस को नहीं पहचानते हैं, वे लेयर 2 एड्रेस को पहचानने में सक्षम हैं, जैसे कि ईथरनेट एड्रेस। ध्यान दें कि होस्ट सभी पाँच परतों को लागू करते हैं; यह इस दृष्टिकोण के अनुरूप है कि इंटरनेट आर्किटेक्चर अपनी अधिकांश जटिलता नेटवर्क के किनारों पर रखता है।

चित्र 1.24 एनकैप्सुलेशन की महत्वपूर्ण अवधारणा को भी दर्शाता है । भेजने वाले होस्ट पर, एक एप्लीकेशन-लेयर संदेश (**चित्र** 1.24 में M) ट्रांसपोर्ट लेयर को भेजा जाता है। सबसे सरल मामले में, ट्रांसपोर्ट लेयर संदेश लेता है और अतिरिक्त जानकारी जोड़ता है (तथाकथित ट्रांसपोर्ट-लेयर

हेडर सूचना, चित्र 1.24 में H) जिसका उपयोग रिसीवर-साइड ट्रांसपोर्ट लेयर द्वारा किया जाएगा। एप्लीकेशन-लेयर संदेश और ट्रांसपोर्ट-लेयर हेडर सूचना मिलकर ट्रांसपोर्ट-लेयर सेगमेंट बनाते हैं। इस प्रकार ट्रांसपोर्ट-लेयर सेगमेंट एप्लीकेशन-लेयर संदेश को समाहित करता है। जोड़ी गई सूचना में रिसीवर-साइड ट्रांसपोर्ट लेयर को संदेश को उचित एप्लिकेशन तक पहुंचाने की अनुमति देने वाली सूचना और त्रुटि-पहचान बिट्स शामिल हो सकते हैं जो रिसीवर को यह निर्धारित करने की अनुमति देते हैं कि संदेश में बिट्स को रूट में बदला गया है या नहीं। ट्रांसपोर्ट लेयर तब सेगमेंट को पास करता है

नेटवर्क लेयर में, जो नेटवर्क-लेयर हेडर जानकारी (चित्र 1.24 में H) जैसे स्रोत और गंतव्य अंत सिस्टम पते जोड़ता है, एक नेटवर्क-लेयर डेटाग्राम बनाता है। फिर डेटाग्राम को लिंक लेयर में भेजा जाता है, जो (बेशक!) अपनी खुद की लिंक-लेयर हेडर जानकारी जोड़ेगा और एक लिंक-लेयर फ्रेम बनाएगा। इस प्रकार, हम देखते हैं कि प्रत्येक लेयर पर, एक पैकेट में दो प्रकार के फ़ील्ड होते हैं: हेडर फ़ील्ड और एक पेलोड फ़ील्ड। पेलोड आमतौर पर ऊपर की लेयर से एक पैकेट होता है।

यहाँ एक उपयोगी उदाहरण सार्वजनिक डाक सेवा के माध्यम से एक कॉर्पोरेट शाखा कार्यालय से दूसरे तक एक अंतर-कार्यालय ज्ञापन भेजना है। मान लीजिए कि ऐलिस, जो एक शाखा कार्यालय में है, बॉब को एक ज्ञापन भेजना चाहती है,

जो दूसरे शाखा कार्यालय में है। मेमो एप्लीकेशन-लेयर संदेश के समान है । एलिस मेमो को एक अंतर-कार्यालय लिफाफे में रखती है, जिसके सामने बॉब का नाम और विभाग लिखा होता है।

अंतर -कार्यालय लिफाफा ट्रांसपोर्ट-लेयर खंड के समान है **- इसमें** हेडर जानकारी (बॉब का नाम और विभाग संख्या) शामिल है और यह एप्लिकेशन-लेयर संदेश (ज्ञापन) को समाहित करता है।

जब भेजने वाले शाखा-कार्यालय का मेलरूम अंतर-कार्यालय लिफाफा प्राप्त करता है, तो वह अंतर-कार्यालय लिफाफे को एक अन्य लिफाफे के अंदर रख देता है, जो सार्वजनिक डाक सेवा के माध्यम से भेजने के लिए उपयुक्त होता है।

भेजने वाला मेलरूम भेजने और प्राप्त करने वाले शाखा कार्यालयों का डाक पता भी लिखता है।

डाक लिफ़ाफ़ा। यहाँ, डाक लिफ़ाफ़ा **डेटाग्राम** के समान है - यह ट्रांसपोर्ट-लेयर सेगमेंट (इंटरऑफ़िस लिफ़ाफ़ा) को समाहित करता है, जो मूल संदेश (मेमो) को समाहित करता है। डाक सेवा डाक लिफ़ाफ़े को प्राप्त करने वाले शाखा-कार्यालय मेलरूम में पहुँचाती है। वहाँ, डी-एनकैप्सुलेशन की प्रक्रिया शुरू होती है। मेलरूम इंटरऑफ़िस मेमो को निकालता है और उसे बॉब को भेजता है।

अंततः बॉब लिफाफा खोलता है और ज्ञापन निकालता है।

एनकैप्सुलेशन की प्रक्रिया ऊपर वर्णित प्रक्रिया से कहीं अधिक जटिल हो सकती है। उदाहरण के लिए, एक बड़े संदेश को कई ट्रांसपोर्ट-लेयर सेगमेंट में विभाजित किया जा सकता है (जो स्वयं कई नेटवर्क-लेयर डेटाग्राम में विभाजित हो सकते हैं)। प्राप्त करने वाले छोर पर, ऐसे सेगमेंट को उसके घटक डेटाग्राम से फिर से बनाया जाना चाहिए।

1.6 नेटवर्क पर हमला

इंटरनेट आज कई संस्थानों के लिए मिशन क्रिटिकल बन गया है, जिसमें बड़ी और छोटी कंपनियाँ, विश्वविद्यालय और सरकारी एजेंसियाँ शामिल हैं। कई व्यक्ति अपने कई व्यावसायिक, सामाजिक और व्यक्तिगत गतिविधियों के लिए भी इंटरनेट पर निर्भर हैं। पहनने योग्य और घरेलू उपकरणों सहित अरबों "चीजें" वर्तमान में इंटरनेट से जुड़ी हुई हैं। लेकिन इस सारी उपयोगिता और उत्साह के पीछे एक अंधेरा पक्ष है, एक ऐसा पक्ष जहाँ "बुरे लोग" हमारे इंटरनेट से जुड़े कंप्यूटरों को नुकसान पहुँचाकर, हमारी गोपनीयता का उल्लंघन करके और जिन इंटरनेट सेवाओं पर हम निर्भर हैं उन्हें निष्क्रिय करके हमारे दैनिक जीवन में तबाही मचाने का प्रयास करते हैं।

नेटवर्क सुरक्षा का क्षेत्र इस बारे में है कि कैसे बुरे लोग कंप्यूटर नेटवर्क पर हमला कर सकते हैं और कैसे हम, जो जल्द ही कंप्यूटर नेटवर्किंग के विशेषज्ञ बनने वाले हैं, उन हमलों के खिलाफ नेटवर्क की रक्षा कर सकते हैं, या इससे भी बेहतर, नए आर्किटेक्चर डिज़ाइन कर सकते हैं जो पहले स्थान पर ऐसे हमलों से प्रतिरक्षित हैं। मौजूदा हमलों की आवृत्ति और विविधता के साथ-साथ नए और अधिक विनाशकारी भविष्य के हमलों के खतरे को देखते हुए, नेटवर्क सुरक्षा कंप्यूटर नेटवर्किंग के क्षेत्र में एक केंद्रीय विषय बन गई है। इस पाठ्यपुस्तक की एक विशेषता यह है कि यह नेटवर्क सुरक्षा के मुद्दों को सबसे आगे लाती है।

चूँकि हमारे पास अभी तक कंप्यूटर नेटवर्किंग और इंटरनेट प्रोटोकॉल में विशेषज्ञता नहीं है, इसलिए हम आज की कुछ अधिक प्रचलित सुरक्षा-संबंधी समस्याओं का सर्वेक्षण करके यहाँ शुरू करेंगे। यह आगामी अध्यायों में अधिक ठोस चर्चाओं के लिए हमारी भूख को बढ़ाएगा। इसलिए हम यहाँ केवल यह पूछकर शुरू करते हैं कि क्या गलत हो सकता है? कंप्यूटर नेटवर्क किस तरह से असुरक्षित हैं? आज के समय में हमलों के कुछ अधिक प्रचलित प्रकार क्या हैं?

बुरे लोग इंटरनेट के माध्यम से आपके होस्ट में मैलवेयर डाल सकते हैं

हम डिवाइस को इंटरनेट से जोड़ते हैं क्योंकि हम इंटरनेट से/इंटरनेट को डेटा प्राप्त करना/भेजना चाहते हैं।

इसमें सभी तरह की अच्छी चीजें शामिल हैं, जैसे इंस्टाग्राम पोस्ट, इंटरनेट सर्च रिजल्ट, स्ट्रीमिंग म्यूजिक, वीडियो कॉन्फ्रेंस कॉल, स्ट्रीमिंग मूवीज, इत्यादि। लेकिन, दुर्भाग्य से, इन सभी अच्छी चीजों के साथ दुर्भावनापूर्ण चीजें भी आती हैं - जिन्हें सामूहिक रूप से मैलवेयर के रूप में जाना जाता है - जो हमारे डिवाइस में प्रवेश कर उन्हें संक्रमित भी कर सकती हैं।

एक बार जब मैलवेयर हमारे डिवाइस को संक्रमित कर देता है, तो यह सभी तरह की चालाकी कर सकता है, जिसमें हमारी फ़ाइलों को हटाना और स्पाइवेयर इंस्टॉल करना शामिल है जो हमारी निजी जानकारी, जैसे कि सोशल सिक्योरिटी नंबर, पासवर्ड और कीस्ट्रोक्स को इकट्ठा करता है, और फिर इसे (बेशक, इंटरनेट के ज़रिए!) बुरे लोगों को वापस भेज देता है। हमारा समझौता किया गया होस्ट हज़ारों समान रूप से समझौता किए गए डिवाइस के नेटवर्क में भी शामिल हो सकता है, जिसे सामूहिक रूप से बॉटनेट के रूप में जाना जाता है, जिसे बुरे लोग स्पैम ई-मेल वितरण या लक्षित होस्ट के खिलाफ़ वितरित इनकार-सेवा हमलों (जल्द ही चर्चा की जाएगी) के लिए नियंत्रित और लाभ उठाते हैं।

आज के समय में मौजूद ज़्यादातर मैलवेयर खुद को दोहराने वाले होते हैं: एक बार जब यह एक होस्ट को संक्रमित कर देता है, तो उस होस्ट से यह इंटरनेट के ज़रिए दूसरे होस्ट में प्रवेश की कोशिश करता है, और नए संक्रमित होस्ट से यह और भी होस्ट में प्रवेश की कोशिश करता है। इस तरह से, खुद को दोहराने वाला मैलवेयर बहुत तेज़ी से फैल सकता है। मैलवेयर वायरस या वर्म के रूप में फैल सकता है। वायरस मैलवेयर होते हैं जिन्हें उपयोगकर्ता के डिवाइस को संक्रमित करने के लिए किसी न किसी तरह के उपयोगकर्ता इंटरैक्शन की ज़रूरत होती है। इसका सबसे बेहतरीन उदाहरण एक ईमेल अटैचमेंट है जिसमें दुर्भावनापूर्ण निष्पादन योग्य कोड होता है। अगर

जब कोई उपयोगकर्ता ऐसा अनुलग्नक प्राप्त करता है और उसे खोलता है, तो उपयोगकर्ता अनजाने में डिवाइस पर मैलवेयर चला देता है।

आमतौर पर, ऐसे ई-मेल वायरस स्वयं-प्रतिकृति होते हैं: एक बार क्रियान्वित होने के बाद, वायरस एक समान दुर्भावनापूर्ण अनुलग्नक के साथ एक समान संदेश भेज सकता है, उदाहरण के लिए, उपयोगकर्ता के पते में प्रत्येक प्राप्तकर्ता को

बुक करें। वर्म्स मैलवेयर हैं जो किसी भी स्पष्ट उपयोगकर्ता सहभागिता के बिना डिवाइस में प्रवेश कर सकते हैं। उदाहरण के लिए,

हो सकता है कि उपयोगकर्ता कोई असुरक्षित नेटवर्क एप्लीकेशन चला रहा हो, जिस पर हमलावर मैलवेयर भेज सकता है।

मामलों में, बिना किसी उपयोगकर्ता के हस्तक्षेप के, एप्लिकेशन इंटरनेट से मैलवेयर स्वीकार कर सकता है और इसे चला सकता है, जिससे एक कीड़ा बन सकता है। नए संक्रमित डिवाइस में कीड़ा फिर इंटरनेट को स्कैन करता है, उसी असुरक्षित नेटवर्क एप्लिकेशन को चलाने वाले अन्य होस्ट की खोज करता है। जब उसे अन्य असुरक्षित होस्ट मिलते हैं, तो वह उन होस्ट को अपनी एक प्रति भेजता है। आज, मैलवेयर व्यापक है और इससे बचाव करना महंगा है। जैसा कि आप इस पाठ्यपुस्तक के माध्यम से काम करते हैं, हम आपको निम्नलिखित प्रश्न के बारे में सोचने के लिए प्रोत्साहित करते हैं: मैलवेयर हमलों से इंटरनेट से जुड़ी डिवाइसों की रक्षा के लिए कंप्यूटर नेटवर्क डिज़ाइनर क्या कर सकते हैं?

बुरे लोग सर्वर और नेटवर्क इंफ्रास्ट्रक्चर पर हमला कर सकते हैं

सुरक्षा खतरों के एक अन्य व्यापक वर्ग को डेनियल-ऑफ-सर्विस (DoS) हमलों के रूप में जाना जाता है। जैसा कि नाम से पता चलता है, DoS हमला किसी नेटवर्क, होस्ट या बुनियादी ढांचे के अन्य हिस्से को वैध रूप से उपयोग करने योग्य नहीं बनाता है।

वेब सर्वर, ई-मेल सर्वर, DNS सर्वर (अध्याय 2 में चर्चा की गई) और संस्थागत नेटवर्क सभी DoS हमलों के अधीन हो सकते हैं। इंटरनेट DoS हमले बेहद आम हैं, जिनमें हज़ारों DoS हमले होते हैं

हर साल होने वाले हमले [मूर 2001]। साइट डिजिटल अटैक मैप हमें दुनिया भर में शीर्ष दैनिक DoS हमलों को देखने की अनुमति देता है [DAM 2016]। अधिकांश इंटरनेट DoS हमले तीन श्रेणियों में से एक में आते हैं:

- भेद्यता हमला। इसमें लक्षित होस्ट पर चल रहे किसी भेद्य एप्लिकेशन या ऑपरेटिंग सिस्टम को कुछ अच्छी तरह से तैयार किए गए संदेश भेजना शामिल है। यदि पैकेट का सही क्रम किसी भेद्य एप्लिकेशन या ऑपरेटिंग सिस्टम को भेजा जाता है, तो सेवा बंद हो सकती है या इससे भी बदतर, होस्ट क्रैश हो सकता है।
- बैंडविड्थ फ्लडिंग। हमलावर लक्षित होस्ट को पैकेटों की बाढ़ भेजता है - इतने सारे पैकेट कि लक्ष्य का एक्सेस लिंक अवरुद्ध हो जाता है, जिससे वैध पैकेट पहुंचने से रुक जाते हैं सर्वर.
- कनेक्शन फ्लडिंग। हमलावर लक्ष्य होस्ट पर बड़ी संख्या में आधे-खुले या पूरी तरह से खुले TCP कनेक्शन (TCP कनेक्शन की चर्चा अध्याय 3 में की गई है) स्थापित करता है। होस्ट इन फर्जी कनेक्शनों से इतना उलझ सकता है कि वह वैध कनेक्शन स्वीकार करना बंद कर देता है।

आइए अब बैंडविड्थ-फ्लडिंग हमले को और विस्तार से देखें। हमारे विलंब और हानि विश्लेषण को याद करते हुए

धारा 1.4.2 में चर्चा से यह स्पष्ट है कि यदि सर्वर की पहुँच दर R bps है, तो हमलावर को नुकसान पहुँचाने के लिए **लगभग R bps की दर से ट्रैफ़िक भेजने की आवश्यकता होगी। यदि** R बहुत बड़ा है, तो एक भी हमला स्रोत सर्वर को नुकसान पहुँचाने के लिए पर्याप्त ट्रैफ़िक उत्पन्न करने में सक्षम नहीं हो सकता है। इसके अलावा, यदि सभी

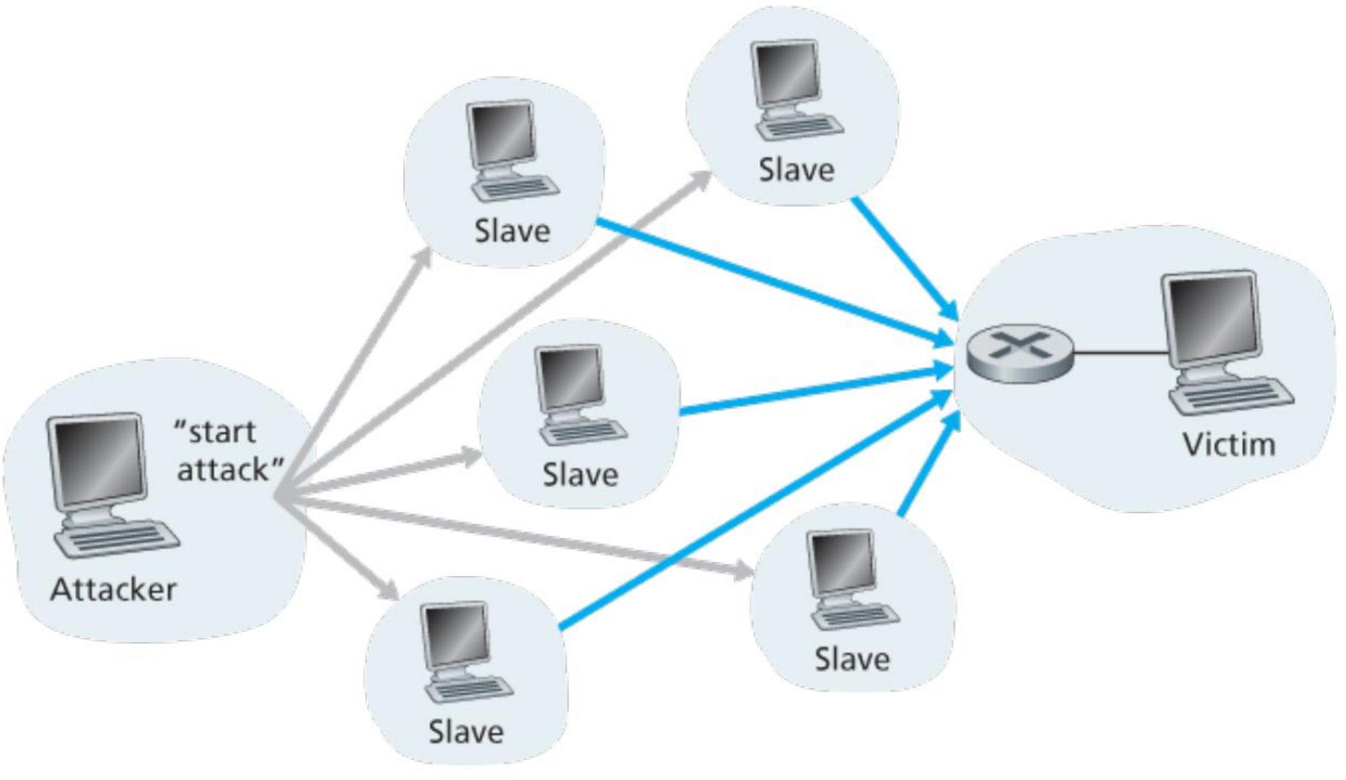
ट्रैफ़िक एक ही स्रोत से निकलता है, तो अपस्ट्रीम राउटर हमले का पता लगाने और ट्रैफ़िक के सर्वर के पास पहुँचने से पहले उस स्रोत से आने वाले सभी ट्रैफ़िक को ब्लॉक करने में सक्षम हो सकता है।

वितरित DoS (DDoS) हमले में,

जैसा कि चित्र 1.25 में दर्शाया गया है , हमलावर कई स्रोतों को नियंत्रित करता है और प्रत्येक स्रोत से लक्ष्य पर ट्रैफ़िक को विस्फोटित करता है। इस दृष्टिकोण के साथ, सभी नियंत्रित स्रोतों में कुल ट्रैफ़िक दर को नियंत्रित करने की आवश्यकता है

सेवा को पंगु बनाने के लिए लगभग R. DDoS हमले हज़ारों होस्ट वाले बॉटनेट का लाभ उठाते हुए आजकल एक आम घटना है [DAM 2016]। DDoS हमलों का पता लगाना और उनसे बचाव करना एक ही होस्ट से DoS हमले की तुलना में बहुत कठिन है।

हम आपको इस पुस्तक को पढ़ते समय निम्नलिखित प्रश्न पर विचार करने के लिए प्रोत्साहित करते हैं: DoS हमलों से बचाव के लिए कंप्यूटर नेटवर्क डिज़ाइनर क्या कर सकते हैं? हम देखेंगे कि तीन प्रकार के DoS हमलों के लिए अलग-अलग बचाव की आवश्यकता होती है।



चित्र 1.25 वितरित सेवा अस्वीकार हमला

बुरे लोग पैकेट सूँघ सकते हैं

आजकल बहुत से उपयोगकर्ता वायरलेस डिवाइस जैसे कि वाई-फाई से जुड़े लैपटॉप या हैंडहेल्ड डिवाइस के माध्यम से इंटरनेट का उपयोग करते हैं।

सेलुलर इंटरनेट कनेक्शन वाले डिवाइस (अध्याय 7 में शामिल)। जबकि सर्वव्यापी इंटरनेट एक्सेस बेहद सुविधाजनक है और मोबाइल उपयोगकर्ताओं के लिए शानदार नए एप्लिकेशन सक्षम करता है, यह एक बड़ी सुरक्षा भेद्यता भी पैदा करता है - वायरलेस ट्रांसमीटर के आसपास एक निष्क्रिय रिसीवर रखकर, वह रिसीवर प्रसारित होने वाले प्रत्येक पैकेट की एक प्रति प्राप्त कर सकता है! इन पैकेट में पासवर्ड, सामाजिक सुरक्षा नंबर, व्यापार रहस्य और निजी व्यक्तिगत संदेश सहित सभी प्रकार की संवेदनशील जानकारी हो सकती है। एक निष्क्रिय रिसीवर जो हर पैकेट की एक प्रति रिकॉर्ड करता है, उसे पैकेट कहा जाता है

सूँघनेवाला.

स्निफर्स को वायर्ड वातावरण में भी तैनात किया जा सकता है। वायर्ड प्रसारण वातावरण में, जैसा कि कई ईथरनेट LAN में होता है, पैकेट स्निफर LAN पर भेजे गए प्रसारण पैकेट की प्रतियां प्राप्त कर सकता है। जैसा कि वर्णित है

अनुभाग 1.2 में , केबल एक्सेस प्रौद्योगिकियां भी पैकेट प्रसारित करती हैं और इस प्रकार सूँघने के प्रति संवेदनशील होती हैं। इसके अलावा, कोई बुरा आदमी जो किसी संस्थान के एक्सेस राउटर या इंटरनेट एक्सेस लिंक तक पहुंच प्राप्त कर लेता है हो सकता है कि वह एक स्निफर स्थापित कर सके जो संगठन से/संगठन में आने वाले हर पैकेट की एक कॉपी बना ले। फिर स्निफ किए गए पैकेट का संवेदनशील जानकारी के लिए ऑफ़लाइन विश्लेषण किया जा सकता है।

पैकेट-स्निफ़िंग सॉफ़्टवेयर विभिन्न वेब साइटों पर और व्यावसायिक उत्पादों के रूप में मुफ़्त में उपलब्ध है। नेटवर्किंग पाठ्यक्रम पढ़ाने वाले छात्रों को प्रयोगशाला अभ्यास देने के लिए जाना जाता है जिसमें पैकेट लिखना शामिल होता है-

स्निफिंग और एप्लिकेशन-लेयर डेटा रिकंस्ट्रक्शन प्रोग्राम। वास्तव में, इस पाठ से जुड़े वायरशार्क [वायरशार्क 2016] लैब (इस अध्याय के अंत में परिचयात्मक वायरशार्क लैब **देखें**) बिल्कुल ऐसे ही पैकेट स्निफर का उपयोग करते हैं!

क्योंकि पैकेट स्निफर निष्क्रिय होते हैं - यानी, वे चैनल में पैकेट इंजेक्ट नहीं करते हैं - उन्हें पहचानना मुश्किल होता है। इसलिए, जब हम वायरलेस चैनल में पैकेट भेजते हैं, तो हमें इस संभावना को स्वीकार करना चाहिए कि कोई बुरा आदमी हमारे पैकेट की प्रतियां रिकॉर्ड कर रहा हो। जैसा कि आपने अनुमान लगाया होगा, पैकेट स्निफिंग के खिलाफ कुछ बेहतरीन बचाव में क्रिप्टोग्राफी शामिल है। हम क्रिप्टोग्राफी की जांच करेंगे क्योंकि यह किस पर लागू होती है

अध्याय 8 में नेटवर्क सुरक्षा .

बुरे लोग आपके भरोसेमंद व्यक्ति का वेश धारण कर सकते हैं

यह आश्चर्यजनक रूप से आसान है (इस लेख को पढ़ते समय आपको ऐसा करने का ज्ञान हो जाएगा!) मनमाने स्रोत पते, पैकेट सामग्री और गंतव्य पते के साथ एक पैकेट बनाना और फिर इस हस्तनिर्मित पैकेट को इंटरनेट में संचारित करना, जो पैकेट को उसके गंतव्य तक कर्तव्यपूर्वक अग्रेषित करेगा। कल्पना करें कि एक अनजान रिसीवर (मान लीजिए कि एक इंटरनेट राउटर) जो इस तरह का पैकेट प्राप्त करता है, (झूठे) स्रोत पते को सत्य मानता है, और फिर पैकेट की सामग्री में एम्बेडेड कुछ कमांड निष्पादित करता है (मान लीजिए कि इसकी अग्रेषण तालिका को संशोधित करता है)। झूठे स्रोत पते के साथ इंटरनेट में पैकेट को इंजेक्ट करने की क्षमता को आईपी स्पूफिंग के रूप में जाना जाता है, और यह उन कई तरीकों में से एक है जिसमें एक उपयोगकर्ता दूसरे उपयोगकर्ता के रूप में छद्मवेश कर सकता है।

इस समस्या को हल करने के लिए, हमें एंड-पॉइंट प्रमाणीकरण की आवश्यकता होगी, यानी, एक ऐसा तंत्र जो हमें निश्चितता के साथ यह निर्धारित करने की अनुमति देगा कि कोई संदेश उस जगह से आता है जहाँ से हम सोचते हैं कि वह आता है। एक बार फिर, हम आपको यह सोचने के लिए प्रोत्साहित करते हैं कि नेटवर्क अनुप्रयोगों और प्रोटोकॉल के लिए यह कैसे किया जा सकता है क्योंकि आप आगे बढ़ते हैं

इस पुस्तक के अध्यायों को पढ़ें। हम अध्याय 8 में अंतिम-बिंदु प्रमाणीकरण के लिए तंत्र का पता लगाएंगे।

इस खंड को समाप्त करते हुए, यह विचार करना उचित है कि इंटरनेट पहले स्थान पर इतना असुरक्षित स्थान कैसे बन गया। इसका उत्तर, संक्षेप में, यह है कि इंटरनेट को मूल रूप से इस तरह से डिज़ाइन किया गया था, जो कि

"पारदर्शी नेटवर्क से जुड़े परस्पर विश्वास करने वाले उपयोगकर्ताओं के समूह" का मॉडल [ब्लूमेंटल 2001] - एक ऐसा मॉडल जिसमें (परिभाषा के अनुसार) सुरक्षा की कोई आवश्यकता नहीं है। मूल इंटरनेट आर्किटेक्चर के कई पहलू पारस्परिक विश्वास की इस धारणा को गहराई से दर्शाते हैं। उदाहरण के लिए, एक उपयोगकर्ता द्वारा एक संदेश भेजने की क्षमता

किसी अन्य उपयोगकर्ता को पैकेट भेजना अनुरोधित/अनुदानित क्षमता के बजाय डिफ़ॉल्ट है, और उपयोगकर्ता की पहचान को डिफ़ॉल्ट रूप से प्रमाणित करने के बजाय घोषित अंकित मूल्य पर लिया जाता है।

लेकिन आज के इंटरनेट में निश्चित रूप से "परस्पर विश्वास करने वाले उपयोगकर्ता" शामिल नहीं हैं। फिर भी, आज के उपयोगकर्ता अभी भी जब वे एक दूसरे पर भरोसा नहीं करते हैं, तो उन्हें संवाद करने की आवश्यकता होती है, संवाद करने की इच्छा हो सकती है गुमनाम रूप से, तीसरे पक्ष के माध्यम से अप्रत्यक्ष रूप से संवाद कर सकते हैं (उदाहरण के लिए, वेब कैश, जिसका हम अध्ययन करेंगे) अध्याय 2, या गतिशीलता-सहायक एजेंट, जिसका हम अध्याय 7 में अध्ययन करेंगे), और हार्डवेयर, सॉफ़्टवेयर और यहाँ तक कि हवा पर भी अविश्वास कर सकते हैं जिसके माध्यम से वे संचार करते हैं। इस पुस्तक को आगे बढ़ाते हुए अब हमारे सामने कई सुरक्षा-संबंधी चुनौतियाँ हैं: हमें स्निफ़िंग, एंड-पॉइंट मास्करेडिंग, मैन-इन-द-मिडल अटैक, DDoS अटैक, मैलवेयर और बहुत कुछ के खिलाफ़ बचाव की तलाश करनी चाहिए। हमें ध्यान रखना चाहिए

ध्यान रखें कि परस्पर विश्वसनीय उपयोगकर्ताओं के बीच संचार नियम के बजाय अपवाद है। आपका स्वागत है आधुनिक कंप्यूटर नेटवर्किंग की दुनिया में!

1.7 कंप्यूटर नेटवर्किंग और इंटरनेट का इतिहास

सेक्शन 1.1 से 1.6 में कंप्यूटर नेटवर्किंग और इंटरनेट की तकनीक का अवलोकन प्रस्तुत किया गया है। अब आपको अपने परिवार और दोस्तों को प्रभावित करने के लिए पर्याप्त जानकारी होनी चाहिए! हालाँकि, यदि आप वास्तव में अगली कॉकटेल पार्टी में बड़ी हिट बनना चाहते हैं, तो आपको अपने प्रवचन में इसके बारे में छोटी-छोटी बातें बतानी चाहिए।

इंटरनेट का आकर्षक इतिहास [सेगलर 1998].

1.7.1 पैकेट स्विचिंग का विकास: 1961-1972

कंप्यूटर नेटवर्किंग और आज के इंटरनेट का क्षेत्र 1960 के दशक की शुरुआत में शुरू हुआ था।

जब टेलीफोन नेटवर्क दुनिया का प्रमुख संचार नेटवर्क था। सेक्शन 1.3 **से याद करें** कि टेलीफोन नेटवर्क प्रेषक से रिसीवर तक सूचना संचारित करने के लिए सर्किट स्विचिंग का उपयोग करता है - यह एक उचित विकल्प है, क्योंकि प्रेषक और रिसीवर के बीच आवाज एक स्थिर दर पर संचारित होती है। 1960 के दशक की शुरुआत में कंप्यूटर के बढ़ते महत्व और टाइमशेयर्ड कंप्यूटर के आगमन को देखते हुए, यह विचार करना शायद स्वाभाविक था कि कंप्यूटर को एक साथ कैसे जोड़ा जाए ताकि उन्हें आपस में साझा किया जा सके।

भौगोलिक रूप से वितरित उपयोगकर्ता। ऐसे उपयोगकर्ताओं द्वारा उत्पन्न ट्रैफ़िक संभवतः बहुत ज़्यादा होगा - गतिविधि के अंतराल, जैसे कि किसी दूरस्थ कंप्यूटर को कमांड भेजना, उसके बाद उत्तर की प्रतीक्षा करते समय या प्राप्त प्रतिक्रिया पर विचार करते समय निष्क्रियता की अवधि।

दुनिया भर के तीन शोध समूहों, जिनमें से प्रत्येक दूसरे के काम से अनभिज्ञ था [लीनर 1998], ने पैकेट स्विचिंग को सर्किट स्विचिंग के लिए एक कुशल और मजबूत विकल्प के रूप में आविष्कार करना शुरू किया।

पैकेट-स्विचिंग तकनीक पर सबसे पहला काम लियोनार्ड क्लेनरॉक [क्लेनरॉक 1961; क्लेनरॉक 1964] का था, जो उस समय एमआईटी में स्नातक छात्र थे। कतार सिद्धांत का उपयोग करते हुए, क्लेनरॉक के काम ने शानदार ढंग से प्रदर्शित किया

बर्स्टी ट्रैफ़िक स्रोतों के लिए पैकेट-स्विचिंग दृष्टिकोण की प्रभावशीलता। 1964 में, रैंड इंस्टीट्यूट में पॉल बारन [बारन 1964] ने सैन्य नेटवर्क पर सुरक्षित आवाज़ के लिए पैकेट स्विचिंग के उपयोग की जांच शुरू कर दी थी, और इंग्लैंड में नेशनल फिजिकल लेबोरेटरी में, डोनाल्ड डेविस और रोजर स्कैंटलबरी भी पैकेट स्विचिंग पर अपने विचार विकसित कर रहे थे।

एमआईटी, रैंड और एनपीएल में किए गए काम ने आज के इंटरनेट की नींव रखी। लेकिन इंटरनेट का एक लंबा इतिहास भी है जिसमें चलो-इसे-बनाएं-और-इसे-दिखाएं जो 1960 के दशक से चली आ रही है। जेसीआर

लिक्लिडर [दिसम्बर 1990] और लॉरेंस रॉबर्ट्स, जो एमआईटी में क्लेनरॉक के दोनों सहकर्मी थे, ने संयुक्त राज्य अमेरिका में एडवांस्ड रिसर्च प्रोजेक्ट्स एजेंसी (एआरपीए) में कंप्यूटर विज्ञान कार्यक्रम का नेतृत्व किया।

रॉबर्ट्स ने ARPAnet [रॉबर्ट्स 1967] के लिए एक समग्र योजना प्रकाशित की, जो पहला पैकेट-स्विच्ड कंप्यूटर नेटवर्क और आज के सार्वजनिक इंटरनेट का प्रत्यक्ष पूर्वज था। 1969 में मजदूर दिवस पर, क्लेनरॉक की देखरेख में UCLA में पहला पैकेट स्विच स्थापित किया गया था, और तीन अतिरिक्त पैकेट स्विच स्थापित किए गए थे

इसके तुरंत बाद स्टैनफोर्ड रिसर्च इंस्टीट्यूट (एसआरआई), यूसी सांता बारबरा और यूटा विश्वविद्यालय में

(चित्र 1.26)। 1969 के अंत तक इंटरनेट का पहला अग्रदूत चार नोड्स बड़ा था। क्लेनरॉक यूसीएलए से एसआरआई तक रिमोट लॉगिन करने के लिए नेटवर्क के सबसे पहले इस्तेमाल को याद करते हैं, जिससे इंटरनेट क्रैश हो गया था।

प्रणाली [क्लेनरॉक 2004].

1972 तक, ARPAnet लगभग 15 नोड्स तक बढ़ गया था और रॉबर्ट काहन द्वारा इसका पहला सार्वजनिक प्रदर्शन किया गया था। ARPAnet एंड सिस्टम के बीच पहला होस्ट-टू-होस्ट प्रोटोकॉल, जिसे नेटवर्क- के रूप में जाना जाता है।

कंट्रोल प्रोटोकॉल (NCP) का काम पूरा हो गया [RFC 001]। एंड-टू-एंड प्रोटोकॉल उपलब्ध होने के कारण, अब एप्लिकेशन लिखे जा सकते थे। रे टॉमलिंसन ने 1972 में पहला ईमेल प्रोग्राम लिखा था।

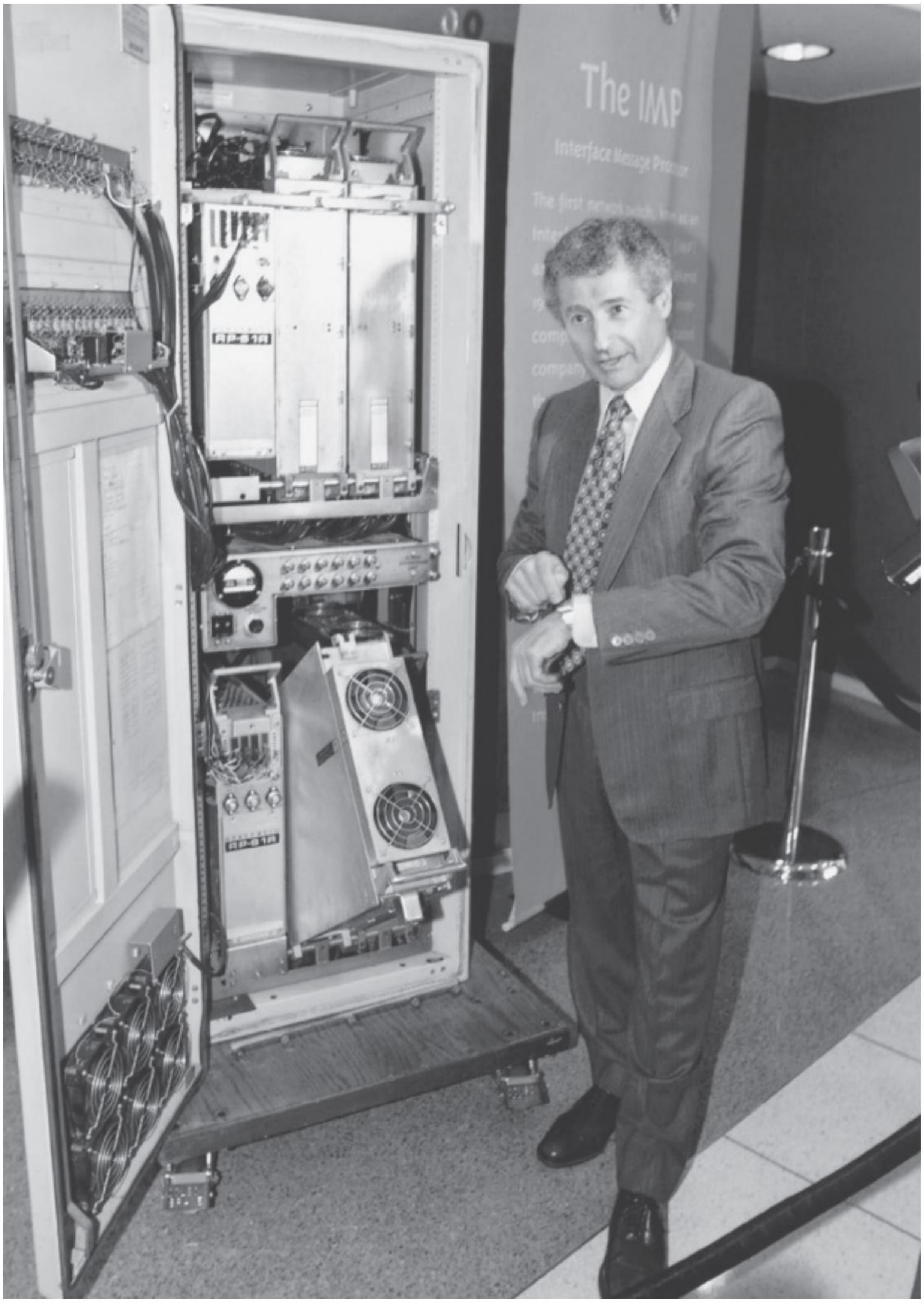
1.7.2 मालिकाना नेटवर्क और इंटरनेटवर्किंग: 1972–1980

प्रारंभिक ARPAnet एक एकल, बंद नेटवर्क था। ARPAnet होस्ट के साथ संचार करने के लिए,

वास्तव में इसे किसी अन्य ARPAnet IMP से जोड़ा जाना था। 1970 के दशक के मध्य से लेकर शुरुआत तक, अतिरिक्त स्टैंड-अलोन

ARPAnet के अलावा पैकेट-स्विचिंग नेटवर्क अस्तित्व में आया: ALOHANet, एक माइक्रोवेव नेटवर्क जो जोड़ता है

हवाई द्वीप पर स्थित विश्वविद्यालय [अब्रामसन 1970], साथ ही DARPA के पैकेट-सैटेलाइट [RFC 829]



चित्र 1.26 एक प्रारंभिक पैकेट स्विच

और पैकेट-रेडियो नेटवर्क [काहन 1978]; टेलीनेट, ARPAnet तकनीक पर आधारित एक BBN वाणिज्यिक पैकेट-स्विचिंग नेटवर्क; साइक्लेड्स, लुई पोज़िन द्वारा अग्रणी एक फ्रांसीसी पैकेट-स्विचिंग नेटवर्क

[थिंक 2012]; टाइम-शेयरिंग नेटवर्क जैसे कि टाइमनेट और जीई सूचना सेवा नेटवर्क, अन्य के अलावा, 1960 के दशक के अंत और 1970 के दशक के प्रारंभ में [श्वार्ट्ज 1977]; आईबीएम का एसएनए (1969-1974), जो एआरपीएनेट कार्य के समानांतर था [श्वार्ट्ज 1977]।

नेटवर्क की संख्या बढ़ रही थी। पीछे मुड़कर देखने पर हम देख सकते हैं कि नेटवर्क को आपस में जोड़ने के लिए एक व्यापक वास्तुकला विकसित करने का समय आ गया था। नेटवर्क को आपस में जोड़ने पर अग्रणी कार्य (डिफेंस एडवांस्ड रिसर्च प्रोजेक्ट्स एजेंसी के प्रायोजन के तहत)

(DARPA)), संक्षेप में नेटवर्कों का एक नेटवर्क **बनाने का काम** विंटन सेर्फ़ और रॉबर्ट काह्न [सेर्फ़] **द्वारा किया गया था**
1974]; इस कार्य का वर्णन करने के लिए इंटरनेटिंग शब्द **गढ़ा गया था।**

ये आर्किटेक्चरल सिद्धांत TCP में सन्निहित थे। हालाँकि, TCP के शुरुआती संस्करण आज के TCP से काफी अलग थे। TCP के शुरुआती संस्करणों ने एंड-सिस्टम रीट्रांसमिशन (आज भी TCP का हिस्सा) के माध्यम से डेटा की विश्वसनीय इन-सीक्वेंस डिलीवरी को अग्रेषित कार्यों (जो आज IP द्वारा किए जाते हैं) के साथ जोड़ा। TCP के साथ शुरुआती प्रयोग, पैकेटाइज्ड वॉयस जैसे अनुप्रयोगों के लिए एक अविश्वसनीय, गैर-प्रवाह-नियंत्रित, एंड-टू-एंड ट्रांसपोर्ट सेवा के महत्व की मान्यता के साथ मिलकर, TCP से IP को अलग करने और UDP प्रोटोकॉल के विकास का नेतृत्व किया। तीन प्रमुख इंटरनेट प्रोटोकॉल जिन्हें हम आज देखते हैं - TCP, UDP और IP - 1960 के दशक के अंत तक वैचारिक रूप से मौजूद थे।

1970 के दशक.

DARPA इंटरनेट से संबंधित शोध के अलावा, कई अन्य महत्वपूर्ण नेटवर्किंग गतिविधियाँ चल रही थीं। हवाई में, नॉर्मन अब्रामसन ALOHAnet विकसित कर रहे थे, जो एक पैकेट-आधारित रेडियो नेटवर्क था जो हवाई द्वीप पर कई दूरस्थ साइटों को एक-दूसरे के साथ संचार करने की अनुमति देता था। ALOHA

प्रोटोकॉल [अब्रामसन 1970] पहला मल्टीपल-एक्सेस प्रोटोकॉल था, जो भौगोलिक रूप से वितरित उपयोगकर्ताओं को एक एकल प्रसारण संचार माध्यम (रेडियो आवृत्ति) साझा करने की अनुमति देता था। मेटकाफ और बोग्स
एब्रामसन के मल्टीपल-एक्सेस प्रोटोकॉल कार्य पर आधारित जब उन्होंने वायर-आधारित साझा प्रसारण नेटवर्क के लिए ईथरनेट प्रोटोकॉल [मेटकाफ 1976] **विकसित किया। दिलचस्प बात यह है कि मेटकाफ और बोग्स का ईथरनेट प्रोटोकॉल कई पीसी, प्रिंटर और साझा डिस्क को जोड़ने की आवश्यकता से प्रेरित था** [पर्किन्स 1994]। पच्चीस साल पहले, पीसी क्रांति और नेटवर्क के विस्फोट से बहुत पहले, मेटकाफ और बोग्स आज के पीसी लैन की नींव रख रहे थे।

1.7.3 नेटवर्क का प्रसार: 1980–1990

1970 के दशक के अंत तक, लगभग दो सौ होस्ट ARPAnet से जुड़े थे। 1980 के दशक के अंत तक सार्वजनिक इंटरनेट से जुड़े होस्ट की संख्या, जो आज के इंटरनेट की तरह दिखने वाले नेटवर्कों का एक संघ है, एक लाख तक पहुँच जाएगी। 1980 का दशक जबरदस्त वृद्धि का समय होगा।

इस वृद्धि का एक बड़ा हिस्सा विश्वविद्यालयों को आपस में जोड़ने वाले कंप्यूटर नेटवर्क बनाने के कई अलग-अलग प्रयासों का नतीजा था। BITNET ने पूर्वोत्तर के कई विश्वविद्यालयों के बीच ईमेल और फ़ाइल ट्रांसफ़र की सुविधा प्रदान की। CSNET (कंप्यूटर साइंस नेटवर्क) का गठन उन विश्वविद्यालय शोधकर्ताओं को जोड़ने के लिए किया गया था, जिनके पास ARPAnet तक पहुँच नहीं थी। 1986 में, NSFNET को NSF-प्रायोजित सुपरकंप्यूटिंग केंद्रों तक पहुँच प्रदान करने के लिए
बनाया गया था।
56 केबीपीएस की प्रारंभिक बैकबोन गति से शुरू होकर, एनएसएफनेट की बैकबोन दशक के अंत तक 1.5 एमबीपीएस पर चलने लगेगी और क्षेत्रीय नेटवर्कों को जोड़ने वाली प्राथमिक बैकबोन के रूप में काम करेगी।

ARPAnet समुदाय में, आज के इंटरनेट आर्किटेक्चर के कई अंतिम टुकड़े अपनी जगह पर आ रहे थे। 1 जनवरी, 1983 को TCP/IP को आधिकारिक तौर पर नए मानक होस्ट प्रोटोकॉल के रूप में लागू किया गया।

ARPAnet (NCP प्रोटोकॉल की जगह)। NCP से TCP/IP में परिवर्तन [RFC 801] एक फ्लैग डे इवेंट था - सभी होस्ट को उस दिन से TCP/IP में स्थानांतरित होना आवश्यक था। 1980 के दशक के अंत में, महत्वपूर्ण

होस्ट-आधारित कंजेशन कंट्रोल को लागू करने के लिए TCP में एक्सटेंशन किए गए [जैकबसन 1988]। DNS, मानव-पठनीय इंटरनेट नाम (उदाहरण के लिए, gaia.cs.umass.edu) और इसके 32-बिट के बीच मैप करने के लिए उपयोग किया जाता है

आईपी पता भी विकसित किया गया था [आरएफसी 1034].

ARPAnet (जो कि अधिकांशतः अमेरिका का प्रयास था) के इस विकास के समानांतर, 1980 के दशक की शुरुआत में फ्रांस ने मिनिटेल परियोजना शुरू की, जो हर किसी के घर में डेटा नेटवर्किंग लाने की एक महत्वाकांक्षी योजना थी। फ्रांसीसी सरकार द्वारा प्रायोजित, मिनिटेल प्रणाली में एक सार्वजनिक पैकेट-स्विच्ड नेटवर्क (X.25 प्रोटोकॉल सूट पर आधारित), मिनिटेल सर्वर और बिल्ट-इन लो-स्पीड मोडेम के साथ सस्ते टर्मिनल शामिल थे। मिनिटेल 1984 में एक बड़ी सफलता बन गई जब फ्रांसीसी सरकार ने प्रत्येक फ्रांसीसी घराने को एक निःशुल्क मिनिटेल टर्मिनल दिया जो इसे चाहता था। मिनिटेल साइटों में निःशुल्क साइटें शामिल थीं - जैसे कि एक टेलीफोन निर्देशिका साइट - साथ ही निजी साइटें, जो प्रत्येक उपयोगकर्ता से उपयोग-आधारित शुल्क एकत्र करती थीं। 1990 के दशक के मध्य में अपने चरम पर, इसने 20,000 से अधिक सेवाएँ प्रदान कीं, जिनमें होम बैंकिंग से लेकर विशेष शोध डेटाबेस तक शामिल थे। मिनिटेल फ्रांसीसी घरों के एक बड़े हिस्से में 10 साल पहले से था जब अधिकांश अमेरिकियों ने इंटरनेट के बारे में कभी सुना भी नहीं था।

1.7.4 इंटरनेट विस्फोट: 1990 का दशक

1990 के दशक में कई ऐसी घटनाएं हुईं जो इंटरनेट के निरंतर विकास और जल्द ही आने वाले व्यावसायीकरण का प्रतीक थीं। इंटरनेट के जनक ARPAnet का अस्तित्व समाप्त हो गया। 1991 में, NSFNET ने वाणिज्यिक उद्देश्यों के लिए NSFNET के उपयोग पर अपने प्रतिबंध हटा लिए। 1995 में NSFNET को बंद कर दिया गया, और इंटरनेट बैकबोन ट्रैफ़िक को वाणिज्यिक द्वारा संचालित किया जाने लगा।

इंटरनेट सेवा प्रदाता.

1990 के दशक की मुख्य घटना वर्ल्ड वाइड वेब एप्लीकेशन का उदय होना था, जिसने इंटरनेट को दुनिया भर के लाखों लोगों के घरों और व्यवसायों तक पहुँचाया। वेब ने सैकड़ों नए अनुप्रयोगों को सक्षम और तैनात करने के लिए एक मंच के रूप में कार्य किया, जिन्हें हम आज के समय में सामान्य मानते हैं, जिसमें खोज (जैसे, गूगल और बिंग) इंटरनेट कॉमर्स (जैसे, अमेज़ॅन और ईबे) और सोशल नेटवर्क (जैसे, फेसबुक) शामिल हैं।

वेब का आविष्कार सर्न में टिम बर्नर्स-ली द्वारा 1989 और 1991 के बीच किया गया था [बर्नर्स-ली 1989], जो 1940 के दशक में वेन्नेवर बुश [बुश 1945] **द्वारा हाइपरटेक्स्ट पर किए गए पहले के काम से उत्पन्न विचारों पर आधारित था।**

और 1960 के दशक से टेड नेल्सन [ज़ानाडू 2012] द्वारा। बर्नर्स-ली और उनके सहयोगियों ने HTML, HTTP, एक वेब सर्वर और एक ब्राउज़र के शुरुआती संस्करण विकसित किए - जो वेब के चार प्रमुख घटक हैं। 1993 के अंत में लगभग दो सौ वेब सर्वर संचालन में थे, सर्वरों का यह संग्रह

यह बस आने वाली बात का एक संकेत था। लगभग इसी समय कई शोधकर्ता GUI इंटरफेस के साथ वेब ब्राउज़र विकसित कर रहे थे, जिनमें मार्क एंड्रीसेन भी शामिल थे, जिन्होंने जिम क्लार्क के साथ मिलकर GUI ब्राउज़र बनाया था।

मोज़ेक कम्युनिकेशंस, जो बाद में नेटस्केप कम्युनिकेशंस कॉर्पोरेशन बन गया [कुसुमानो 1998; क्विटनर 1998]। 1995 तक, विश्वविद्यालय के छात्र दैनिक आधार पर वेब सर्फ करने के लिए नेटस्केप ब्राउज़र का उपयोग कर रहे थे। लगभग इसी समय बड़ी और छोटी कंपनियों ने वेब सर्वर संचालित करना और लेनदेन करना शुरू कर दिया

वेब पर वाणिज्य। 1996 में, माइक्रोसॉफ्ट ने ब्राउज़र बनाना शुरू किया, जिससे ब्राउज़र युद्ध शुरू हो गया

नेटस्केप और माइक्रोसॉफ्ट के बीच, जिसे माइक्रोसॉफ्ट ने कुछ साल बाद जीत लिया [कुसुमानो 1998]।

1990 के दशक का दूसरा भाग इंटरनेट के लिए जबरदस्त विकास और नवाचार का दौर था, जिसमें प्रमुख निगमों और हजारों स्टार्टअप ने इंटरनेट उत्पाद और सेवाएँ बनाईं। सहस्राब्दी के अंत तक इंटरनेट सैकड़ों लोकप्रिय अनुप्रयोगों का समर्थन कर रहा था, जिनमें चार किलर अनुप्रयोग शामिल थे:

- ई-मेल, अनुलग्नकों और वेब-सुलभ ई-मेल सहित
- वेब, जिसमें वेब ब्राउज़िंग और इंटरनेट वाणिज्य शामिल हैं
- संपर्क सूची के साथ त्वरित संदेश
- एमपी3 की पीयर-टू-पीयर फ़ाइल शेयरिंग, नैपस्टर द्वारा अग्रणी

दिलचस्प बात यह है कि पहले दो किलर एप्लीकेशन अनुसंधान समुदाय से आए थे, जबकि अंतिम दो कुछ युवा उद्यमियों द्वारा बनाए गए थे।

1995 से 2001 तक का समय वित्तीय बाज़ारों में इंटरनेट के लिए उतार-चढ़ाव भरा रहा। मुनाफ़े में आने से पहले ही सैकड़ों इंटरनेट स्टार्टअप ने आरंभिक सार्वजनिक पेशकश की और शेयर बाज़ार में कारोबार करना शुरू कर दिया। कई कंपनियों का मूल्यांकन अरबों डॉलर में था, लेकिन उनके पास कोई ख़ास राजस्व स्रोत नहीं था। 2000-2001 में इंटरनेट स्टॉक गिर गए और कई स्टार्टअप बंद हो गए।

फिर भी, इंटरनेट क्षेत्र में कई कंपनियां बड़ी विजेता बनकर उभरीं, जिनमें माइक्रोसॉफ्ट, सिस्को, याहू, ई-बे, गूगल और अमेज़न शामिल हैं।

1.7.5 नई सहस्राब्दी

कंप्यूटर नेटवर्किंग में नवाचार तेजी से जारी है। सभी मोर्चों पर प्रगति हो रही है, जिसमें एक्सेस नेटवर्क और नेटवर्क बैकबोन दोनों में तेज़ राउटर और उच्च संचरण गति की तैनाती शामिल है। लेकिन निम्नलिखित विकास विशेष ध्यान देने योग्य हैं:

- सहस्राब्दी की शुरुआत से ही हम ब्रॉडबैंड की आक्रामक तैनाती देख रहे हैं
  घरों तक इंटरनेट की पहुंच - न केवल केबल मॉडेम और डीएसएल बल्कि घर तक फाइबर भी, जैसा कि चर्चा की गई है
  अनुभाग 1.2 में । इस हाई-स्पीड इंटरनेट एक्सेस ने कई वीडियो अनुप्रयोगों के लिए मंच तैयार किया है, जिसमें उपयोगकर्ता द्वारा निर्मित वीडियो (उदाहरण के लिए, यूट्यूब) का वितरण, फिल्मों और टेलीविज़न शो की ऑन-डिमांड स्ट्रीमिंग (जैसे, नेटफ्लिक्स), और मल्टी-पर्सन वीडियो कॉन्फ्रेंस (जैसे, स्काइप,

फेसटाइम और गूगल हैंगआउट)।

- हाई-स्पीड (54 एमबीपीएस और उससे अधिक) सार्वजनिक वाईफ़ाई नेटवर्क और 4 जी सेलुलर टेलीफोनी नेटवर्क के माध्यम से मध्यम गति (दसियों एमबीपीएस) इंटरनेट एक्सेस की बढ़ती सर्वव्यापकता न केवल चलते-फिरते लगातार जुड़े रहना संभव बना रही है, बल्कि येल्प, टिंडर, यिक याक और वाज़ जैसे नए स्थान-विशिष्ट अनुप्रयोगों को भी सक्षम बना रही है। 2011 में इंटरनेट से कनेक्ट होने वाले वायरलेस उपकरणों की संख्या वायर्ड उपकरणों की संख्या से अधिक हो गई। इस हाई-स्पीड वायरलेस एक्सेस ने हाथ में पकड़े जाने वाले कंप्यूटर (आईफ़ोन, एंड्रॉइड, आईपैड, इत्यादि) के तेज़ी से उभरने के लिए मंच तैयार किया है, जो इंटरनेट तक निरंतर और बिना किसी बाधा के पहुँच का आनंद लेते हैं।

- ऑनलाइन सोशल नेटवर्क - जैसे कि फेसबुक, इंस्टाग्राम, ट्विटर और वीचैट (चीन में बेहद लोकप्रिय) - ने इंटरनेट के शीर्ष पर विशाल लोगों के नेटवर्क बनाए हैं। इनमें से कई सोशल नेटवर्क का इस्तेमाल मैसेजिंग के साथ-साथ फोटो शेयरिंग के लिए भी बड़े पैमाने पर किया जाता है। आज कई इंटरनेट उपयोगकर्ता मुख्य रूप से एक या अधिक सोशल नेटवर्क के भीतर "रहते" हैं। अपने API के माध्यम से, ऑनलाइन सोशल नेटवर्क नए नेटवर्क वाले एप्लिकेशन और वितरित गेम के लिए प्लेटफ़ॉर्म बनाते हैं।

- जैसा कि खंड 1.3.3 में चर्चा की गई है , ऑनलाइन सेवा प्रदाताओं, जैसे कि गूगल और माइक्रोसॉफ्ट ने अपने स्वयं के व्यापक निजी नेटवर्क तैनात किए हैं, जो न केवल उनके वैश्विक रूप से वितरित डेटा केंद्रों को आपस में जोड़ते हैं, बल्कि निचले स्तर के आईएसपी के साथ सीधे जुड़कर जितना संभव हो सके इंटरनेट को बायपास करने के लिए उपयोग किए जाते हैं। नतीजतन, गूगल खोज परिणाम और ई-मेल एक्सेस लगभग तुरंत प्रदान करता है, जैसे कि उनके डेटा सेंटर किसी के अपने कंप्यूटर के भीतर चल रहे हों।

- कई इंटरनेट कॉमर्स कंपनियाँ अब अपने एप्लिकेशन को "क्लाउड" में चला रही हैं - जैसे कि Amazon के EC2 में, Google के एप्लीकेशन इंजन में, या Microsoft के Azure में। कई कंपनियों और विश्वविद्यालयों ने भी अपने इंटरनेट एप्लिकेशन (जैसे, ईमेल और वेब होस्टिंग) को क्लाउड पर माइग्रेट कर दिया है। क्लाउड कंपनियाँ न केवल एप्लिकेशन को स्केलेबल कंप्यूटिंग और स्टोरेज वातावरण प्रदान करती हैं, बल्कि एप्लिकेशन को अपने उच्च-प्रदर्शन वाले निजी नेटवर्क तक अंतर्निहित पहुँच भी प्रदान करती हैं।

1.8 सारांश

इस अध्याय में हमने बहुत सारी सामग्री को कवर किया है! हमने हार्डवेयर और सॉफ़्टवेयर के विभिन्न हिस्सों को देखा है जो विशेष रूप से इंटरनेट और सामान्य रूप से कंप्यूटर नेटवर्क बनाते हैं। हमने नेटवर्क के किनारे से शुरुआत की, अंतिम सिस्टम और अनुप्रयोगों को देखा, और अंतिम सिस्टम पर चलने वाले अनुप्रयोगों को प्रदान की जाने वाली परिवहन सेवा को देखा। हमने लिंक-लेयर तकनीकों और भौतिक मीडिया को भी देखा जो आम तौर पर एक्सेस नेटवर्क में पाए जाते हैं। फिर हमने नेटवर्क के अंदर, नेटवर्क कोर में गहराई से गोता लगाया, पैकेट स्विचिंग और सर्किट स्विचिंग को दूरसंचार नेटवर्क के माध्यम से डेटा परिवहन के लिए दो बुनियादी तरीकों के रूप में पहचाना, और हमने प्रत्येक दृष्टिकोण की ताकत और कमजोरियों की जांच की। हमने वैश्विक इंटरनेट की संरचना की भी जांच की, यह सीखते हुए कि इंटरनेट नेटवर्क का एक नेटवर्क है। हमने देखा कि इंटरनेट की पदानुक्रमित संरचना, जिसमें उच्च-

और निचले स्तर के आईएसपी को शामिल करने के लिए, इसे हजारों नेटवर्क को शामिल करने की अनुमति दी गई है।

इस परिचयात्मक अध्याय के दूसरे भाग में, हमने कंप्यूटर नेटवर्किंग के क्षेत्र के लिए केंद्रीय कई विषयों की जांच की। हमने सबसे पहले पैकेट-स्विच्ड नेटवर्क में देरी, थ्रूपुट और पैकेट हानि के कारणों की जांच की। हमने ट्रांसमिशन, प्रसार और कतारबद्ध देरी के साथ-साथ थ्रूपुट के लिए सरल मात्रात्मक मॉडल विकसित किए; हम इस पुस्तक में होमवर्क समस्याओं में इन देरी मॉडल का व्यापक उपयोग करेंगे। इसके बाद हमने प्रोटोकॉल लेयरिंग और सर्विस मॉडल की जांच की, नेटवर्किंग में प्रमुख वास्तुशिल्प सिद्धांत जिन्हें हम इस पुस्तक में भी संदर्भित करेंगे। हमने इंटरनेट के दिनों में कुछ अधिक प्रचलित सुरक्षा हमलों का भी सर्वेक्षण किया। हमने कंप्यूटर नेटवर्किंग के संक्षिप्त इतिहास के साथ नेटवर्किंग के लिए अपना परिचय समाप्त किया। पहला अध्याय अपने आप में कंप्यूटर नेटवर्किंग में एक मिनी-कोर्स का गठन करता है।

तो, हमने इस पहले अध्याय में वाकई बहुत कुछ कवर किया है! अगर आप थोड़ा अभिभूत हैं, तो चिंता न करें। अगले अध्यायों में हम इन सभी विचारों पर फिर से विचार करेंगे, और उन्हें और अधिक विस्तार से कवर करेंगे (यह एक वादा है, धमकी नहीं!)। इस बिंदु पर, हम आशा करते हैं कि आप इस अध्याय को नेटवर्क बनाने वाले भागों के लिए अभी भी विकसित हो रहे अंतर्ज्ञान, नेटवर्किंग की शब्दावली पर अभी भी विकसित हो रहे नियंत्रण (इस अध्याय को फिर से संदर्भित करने में संकोच न करें), और नेटवर्किंग के बारे में अधिक जानने की बढ़ती इच्छा के साथ छोड़ेंगे। इस पुस्तक के बाकी हिस्सों के लिए यही कार्य हमारे सामने है।

इस पुस्तक का रोडमैप

किसी भी यात्रा को शुरू करने से पहले, आपको हमेशा रोड मैप पर नज़र डालनी चाहिए ताकि आगे आने वाली प्रमुख सड़कों और जंक्शनों से परिचित हो सकें। जिस यात्रा पर हम जाने वाले हैं, उसके लिए अंतिम लक्ष्य कंप्यूटर नेटवर्क के कैसे, क्या और क्यों की गहरी समझ है। हमारा रोड मैप है

इस पुस्तक के अध्यायों का क्रम:

1. कंप्यूटर नेटवर्क और इंटरनेट 2. एप्लीकेशन लेयर 3. ट्रांसपोर्ट लेयर

4. नेटवर्क लेयर: डेटा प्लेन 5. नेटवर्क

लेयर: कंट्रोल प्लेन 6. लिंक लेयर

और LAN 7. वायरलेस और मोबाइल नेटवर्क

8. कंप्यूटर नेटवर्क में सुरक्षा 9. मल्टीमीडिया नेटवर्किंग

अध्याय 2 से 6 इस पुस्तक के पाँच मुख्य अध्याय हैं। आपको ध्यान देना चाहिए कि ये अध्याय पाँच-परत इंटरनेट प्रोटोकॉल की शीर्ष चार परतों के इर्द-गिर्द व्यवस्थित हैं। इसके अलावा ध्यान दें कि हमारी यात्रा इंटरनेट प्रोटोकॉल स्टैक के शीर्ष पर शुरू होगी, अर्थात्, एप्लिकेशन परत, और आगे बढ़ेगी

नीचे की ओर। इस ऊपर से नीचे की यात्रा के पीछे तर्क यह है कि एक बार जब हम अनुप्रयोगों को समझ लेते हैं, तो हम इन अनुप्रयोगों को समर्थन देने के लिए आवश्यक नेटवर्क सेवाओं को समझ सकते हैं। फिर, हम बदले में, उन विभिन्न तरीकों की जांच करें जिनसे ऐसी सेवाओं को नेटवर्क आर्किटेक्चर द्वारा कार्यान्वित किया जा सकता है। इस प्रकार आवेदनों को पहले ही कवर करने से शेष पाठ के लिए प्रेरणा मिलती है।

पुस्तक का दूसरा भाग - अध्याय 7 से 9 - आधुनिक कंप्यूटर नेटवर्किंग में तीन अत्यंत महत्वपूर्ण (और कुछ हद तक स्वतंत्र) विषयों पर प्रकाश डालता है। अध्याय 7 में, हम वायरलेस और मोबाइल नेटवर्क, जिसमें वायरलेस LAN (वाई-फाई और ब्लूटूथ सहित), सेलुलर टेलीफोनी नेटवर्क शामिल हैं, की जांच करते हैं।

(जीएसएम, 3जी और 4जी सहित), और गतिशीलता (आईपी और जीएसएम नेटवर्क दोनों में)। अध्याय 8, जो कंप्यूटर नेटवर्क में सुरक्षा को संबोधित करता है, सबसे पहले एन्क्रिप्शन और नेटवर्क सुरक्षा के आधार को देखता है, और फिर हम जांच करते हैं कि बुनियादी सिद्धांत को इंटरनेट संदर्भों की एक विस्तृत श्रृंखला में कैसे लागू किया जा रहा है। अंतिम अध्याय 8, जो कंप्यूटर नेटवर्क में सुरक्षा को संबोधित करता है, सबसे पहले एन्क्रिप्शन और नेटवर्क सुरक्षा के आधार को देखता है, और फिर हम जांच करते हैं कि बुनियादी सिद्धांत को इंटरनेट संदर्भों की एक विस्तृत श्रृंखला में कैसे लागू किया जा रहा है।

अध्याय, जो मल्टीमीडिया नेटवर्किंग को संबोधित करता है, ऑडियो और वीडियो अनुप्रयोगों की जांच करता है जैसे इंटरनेट फोन, वीडियो कॉन्फ्रेंसिंग और संग्रहीत मीडिया की स्ट्रीमिंग। हम यह भी देखते हैं कि ऑडियो और वीडियो को सेवा की निरंतर गुणवत्ता प्रदान करने के लिए पैकेट-स्विच्ड नेटवर्क को कैसे डिज़ाइन किया जा सकता है अनुप्रयोग.

गृहकार्य की समस्याएं और प्रश्न

अध्याय 1 समीक्षा प्रश्न

अनुभाग 1.1

R1. होस्ट और एंड सिस्टम में क्या अंतर है? एंड सिस्टम के कई अलग-अलग प्रकारों की सूची बनाएँ। क्या वेब सर्वर एक एंड सिस्टम है?

R2. प्रोटोकॉल शब्द का इस्तेमाल अक्सर राजनयिक संबंधों का वर्णन करने के लिए किया जाता है। विकिपीडिया राजनयिक प्रोटोकॉल का वर्णन कैसे करता है?

R3. प्रोटोकॉल के लिए मानक क्यों महत्वपूर्ण हैं?

अनुभाग 1.2

R4. छह एक्सेस तकनीकें सूचीबद्ध करें। प्रत्येक को होम एक्सेस, एंटरप्राइज़ एक्सेस या वाइड-एरिया वायरलेस एक्सेस के रूप में वर्गीकृत करें।

R5. क्या HFC संचरण दर उपयोगकर्ताओं के बीच समर्पित या साझा है? क्या डाउनस्ट्रीम HFC चैनल में टकराव संभव है? क्यों या क्यों नहीं?

R6. अपने शहर में उपलब्ध आवासीय पहुँच तकनीकों की सूची बनाएँ। प्रत्येक प्रकार की पहुँच के लिए, विज्ञापित डाउनस्ट्रीम दर, अपस्ट्रीम दर और मासिक मूल्य प्रदान करें।

R7. ईथरनेट LAN की संचरण दर क्या है?

R8. कुछ भौतिक मीडिया क्या हैं जिन पर ईथरनेट चलाया जा सकता है?

R9. डायल-अप मोडेम, HFC, DSL और FTTH सभी का उपयोग आवासीय पहुँच के लिए किया जाता है। इनमें से प्रत्येक पहुँच तकनीक के लिए, संचरण दरों की एक सीमा प्रदान करें और इस पर टिप्पणी करें कि क्या
संचरण दर साझा या समर्पित है।

R10. आज की सबसे लोकप्रिय वायरलेस इंटरनेट एक्सेस तकनीकों का वर्णन करें। उनकी तुलना करें और उनमें अंतर करें।

अनुभाग 1.3

R11. मान लीजिए कि एक प्रेषक होस्ट और एक प्राप्तकर्ता होस्ट के बीच केवल एक पैकेट स्विच है।
भेजने वाले होस्ट और स्विच के बीच तथा स्विच और होस्ट के बीच संचरण दर

प्राप्त करने वाले होस्ट R और R हैं 1 2, क्रमशः। यह मानते हुए कि स्विच स्टोर-एंड-फॉरवर्ड का उपयोग करता है

पैकेट स्विचिंग में, L लंबाई का पैकेट भेजने में कुल अंत-से-अंत विलंब कितना है ? (कतारबद्धता, प्रसार विलंब और प्रसंस्करण विलंब पर ध्यान न दें।)

R12. पैकेट-स्विच्ड नेटवर्क की तुलना में सर्किट-स्विच्ड नेटवर्क का क्या लाभ है? सर्किट-स्विच्ड नेटवर्क में FDM की तुलना में TDM के क्या लाभ हैं?

R13. मान लीजिए कि उपयोगकर्ता 2 एमबीपीएस लिंक साझा करते हैं। साथ ही मान लीजिए कि प्रत्येक उपयोगकर्ता संचार करते समय लगातार 1 एमबीपीएस पर संचार करता है, लेकिन प्रत्येक उपयोगकर्ता केवल 20 प्रतिशत समय ही संचारित करता है। (देखें

अनुभाग 1.3 में सांख्यिकीय मल्टीप्लेक्सिंग की चर्चा।) a. जब सर्किट स्विचिंग का उपयोग किया

जाता है, तो कितने उपयोगकर्ताओं का समर्थन किया जा सकता है? b. इस समस्या के शेष भाग के लिए, मान लीजिए कि पैकेट

स्विचिंग का उपयोग किया जाता है। यदि दो या उससे कम उपयोगकर्ता एक ही समय में संचारित करते हैं, तो लिंक से पहले अनिवार्य रूप से कोई कतारबद्ध विलंब क्यों नहीं होगा? यदि तीन उपयोगकर्ता एक ही समय में संचारित करते हैं, तो कतारबद्ध विलंब क्यों होगा?

c. किसी दिए गए उपयोगकर्ता द्वारा संचारण किए जाने की संभावना ज्ञात कीजिए। d. मान लीजिए

अब तीन उपयोगकर्ता हैं। किसी भी समय, सभी तीन उपयोगकर्ता एक साथ संचारण कर रहे हैं, इसकी संभावना ज्ञात कीजिए। उस समय का अंश ज्ञात कीजिए जिसके दौरान कतार

बढ़ता है.

R14. पदानुक्रम के एक ही स्तर पर दो ISP अक्सर एक दूसरे के साथ क्यों जुड़े रहते हैं? IXP कैसे पैसा कमाता है?

R15. कुछ सामग्री प्रदाताओं ने अपने स्वयं के नेटवर्क बनाए हैं। Google के नेटवर्क का वर्णन करें। सामग्री प्रदाताओं को ये नेटवर्क बनाने के लिए क्या प्रेरित करता है?

अनुभाग 1.4

R16. एक स्रोत होस्ट से गंतव्य होस्ट तक एक पैकेट को एक निश्चित मार्ग पर भेजने पर विचार करें। अंत-से-अंत विलंब में विलंब घटकों की सूची बनाएँ। इनमें से कौन-सी देरी स्थिर है और कौन-सी नहीं परिवर्तनशील हैं?

R17. कंपेनियन वेब साइट पर ट्रांसमिशन वर्सेस प्रोपेगेशन डिले एप्लेट पर जाएँ। उपलब्ध दरों, प्रसार विलंब और पैकेट आकारों में से, ऐसा संयोजन खोजें जिसके लिए प्रेषक पैकेट के पहले बिट के रिसीवर तक पहुँचने से पहले ही संचारण समाप्त कर दे। ऐसा कोई अन्य संयोजन खोजें जिसके लिए पैकेट का पहला बिट प्रेषक के संचारण समाप्त करने से पहले ही रिसीवर तक पहुँच जाए।

R18. 1,000 बाइट्स लंबाई के पैकेट को 2,500 किलोमीटर की दूरी, 2.5108 मीटर/सेकंड की प्रसार गति और 2 एमबीपीएस की संचरण दर वाले लिंक पर प्रसारित होने में कितना समय लगता है? अधिक सामान्यतः, कितना समय लगता है?

दूरी d के लिंक पर प्रसारित होने में **लंबाई** L के पैकेट को कितना समय लगता है , प्रसार गति

s, और संचरण दर R bps? क्या यह देरी पैकेट की लंबाई पर निर्भर करती है? क्या यह देरी संचरण दर पर निर्भर करती है?

R19. मान लीजिए होस्ट A, होस्ट B को एक बड़ी फ़ाइल भेजना चाहता है। होस्ट A से होस्ट B तक के पथ में तीन लिंक हैं, जिनकी दर R1=500 kbps, R2=2 Mbps, और R3=1 Mbps है।

ए. मान लें कि नेटवर्क में कोई अन्य ट्रैफ़िक नहीं है, तो फ़ाइल ट्रांसफ़र के लिए थ्रूपुट क्या है? बी. मान लें कि फ़ाइल 4 मिलियन बाइट्स की है। फ़ाइल आकार को थ्रूपुट से विभाजित करने पर, मोटे तौर पर कितना होगा?

फ़ाइल को होस्ट बी में स्थानांतरित करने में कितना समय लगेगा?

(c) (a) और (b) को दोहराएँ, लेकिन अब R **को** घटाकर 100 kbps2कर दें।

R20. मान लीजिए कि एंड सिस्टम A, एंड सिस्टम B को एक बड़ी फ़ाइल भेजना चाहता है। बहुत उच्च स्तर पर, बताएं कि एंड सिस्टम A फ़ाइल से पैकेट कैसे बनाता है। जब इनमें से कोई पैकेट राउटर पर आता है, तो राउटर पैकेट में मौजूद किस जानकारी का उपयोग करके यह निर्धारित करता है कि पैकेट को किस लिंक पर भेजा जाए? इंटरनेट में पैकेट स्विचिंग एक शहर से दूसरे शहर तक गाड़ी चलाने और रास्ते में दिशा-निर्देश पूछने के समान क्यों है?

R21. साथी वेब साइट पर कतार और हानि एप्लेट पर जाएँ। अधिकतम उत्सर्जन दर और न्यूनतम संचरण दर क्या है? उन दरों के साथ, यातायात की तीव्रता क्या है?

इन दरों के साथ एप्लेट चलाएँ और निर्धारित करें कि पैकेट हानि होने में कितना समय लगता है। फिर प्रयोग को दूसरी बार दोहराएँ और फिर से निर्धारित करें कि पैकेट हानि होने में कितना समय लगता है। क्या मान अलग हैं? क्यों या क्यों नहीं?

अनुभाग 1.5

R22. पाँच ऐसे कार्यों की सूची बनाएँ जिन्हें एक लेयर कर सकता है। क्या यह संभव है कि इनमें से एक (या अधिक) कार्य दो (या अधिक) लेयर्स द्वारा किया जा सके?

R23. इंटरनेट प्रोटोकॉल स्टैक में पाँच परतें कौन सी हैं? इनमें से प्रत्येक परत की मुख्य ज़िम्मेदारियाँ क्या हैं?

R24. एप्लीकेशन-लेयर मैसेज क्या है? ट्रांसपोर्ट-लेयर सेगमेंट क्या है? नेटवर्क-लेयर डेटाग्राम क्या है? लिंक-लेयर फ्रेम क्या है?

R25. इंटरनेट प्रोटोकॉल स्टैक में कौन सी परतें राउटर प्रोसेस करती हैं? कौन सी परतें लिंक-लेयर स्विच प्रोसेस करती हैं? कौन सी परतें होस्ट प्रोसेस करती हैं?

अनुभाग 1.6

R26. वायरस और वर्म में क्या अंतर है?

R27. बताएं कि बॉटनेट कैसे बनाया जा सकता है और इसका उपयोग DDoS हमले के लिए कैसे किया जा सकता है।

R28. मान लीजिए ऐलिस और बॉब कंप्यूटर नेटवर्क पर एक दूसरे को पैकेट भेज रहे हैं। मान लीजिए ट्रूडी खुद को नेटवर्क में इस तरह से रखती है कि वह एलिस द्वारा भेजे गए सभी पैकेट को कैप्चर कर सके और बॉब को जो चाहे भेज सके; वह बॉब द्वारा भेजे गए सभी पैकेट को भी कैप्चर कर सकती है और एलिस को जो चाहे भेज सकती है। ट्रूडी द्वारा इस स्थिति से किए जा सकने वाले कुछ दुर्भावनापूर्ण कामों की सूची बनाएँ।

समस्याएं

P1. स्वचालित टेलर मशीन और बैंक के केंद्रीकृत कंप्यूटर के बीच उपयोग किए जाने वाले एप्लिकेशन-स्तरीय प्रोटोकॉल को डिज़ाइन और वर्णित करें। आपके प्रोटोकॉल को उपयोगकर्ता के कार्ड और पासवर्ड को सत्यापित करने, खाते की शेष राशि (जो केंद्रीकृत कंप्यूटर पर रखी जाती है) की जांच करने और खाते से निकासी (यानी उपयोगकर्ता को पैसे वितरित करने) की अनुमति देनी चाहिए।

आपकी प्रोटोकॉल इकाइयाँ उस बहुत ही सामान्य मामले को संभालने में सक्षम होनी चाहिए जिसमें निकासी को कवर करने के लिए खाते में पर्याप्त पैसा नहीं है। आदान-प्रदान किए गए संदेशों और स्वचालित टेलर मशीन या बैंक के केंद्रीकृत कंप्यूटर द्वारा संदेशों के प्रसारण और प्राप्ति पर की गई कार्रवाई को सूचीबद्ध करके अपना प्रोटोकॉल निर्दिष्ट करें। बिना किसी त्रुटि के सरल निकासी के मामले के लिए अपने प्रोटोकॉल के संचालन को स्केच करें, जैसा कि चित्र में दिखाया गया है।

चित्र 1.2 . अंतर्निहित अंत-से-अंत परिवहन सेवा के बारे में आपके प्रोटोकॉल द्वारा की गई धारणाओं को स्पष्ट रूप से बताएं।

P2. समीकरण 1.1, संचरण दर R के N **लिंक** पर L लंबाई के एक पैकेट को भेजने के अंत-से-अंत विलंब के लिए एक सूत्र देता है। P ऐसे पैकेट को वापस भेजने के लिए इस सूत्र को सामान्यीकृत करें-

एन **लिंक** पर वापस .

P3. एक ऐसे एप्लिकेशन पर विचार करें जो स्थिर दर पर डेटा संचारित करता है (उदाहरण के लिए, प्रेषक **हर** k समय इकाइयों में डेटा की एक N-बिट इकाई **उत्पन्न करता है, जहाँ** k छोटा और स्थिर है)। साथ ही, जब ऐसा एप्लिकेशन शुरू होता है, तो यह अपेक्षाकृत लंबे समय तक चलता रहेगा। अपने उत्तर को संक्षेप में उचित ठहराते हुए निम्नलिखित प्रश्नों के उत्तर दें:

क. पैकेट-स्विच्ड नेटवर्क या सर्किट-स्विच्ड नेटवर्क अधिक उपयुक्त होगा यह आवेदन? क्यों?

b. मान लीजिए कि एक पैकेट-स्विच्ड नेटवर्क का उपयोग किया जाता है और इस नेटवर्क में एकमात्र ट्रैफ़िक ऊपर वर्णित ऐसे अनुप्रयोगों से आता है। इसके अलावा, मान लें कि अनुप्रयोग डेटा दरों का योग प्रत्येक लिंक की क्षमता से कम है। क्या किसी प्रकार के भीड़ नियंत्रण की आवश्यकता है? क्यों?

P4. चित्र 1.13 में सर्किट-स्विच्ड नेटवर्क पर विचार करें । याद रखें कि प्रत्येक लिंक पर 4 सर्किट हैं। चार स्विच A, B, C और D को घड़ी की दिशा में लेबल करें।

a. एक साथ चल रहे कनेक्शनों की अधिकतम संख्या क्या है? इस नेटवर्क में किसी भी एक समय?

b. मान लीजिए कि सभी कनेक्शन स्विच A और C के बीच हैं। एक साथ चल सकने वाले कनेक्शनों की अधिकतम संख्या क्या है?

c. मान लीजिए कि हम स्विच A और C के बीच चार कनेक्शन बनाना चाहते हैं, तथा स्विच B और D के बीच चार अन्य कनेक्शन बनाना चाहते हैं। क्या हम सभी आठ कनेक्शनों को समायोजित करने के लिए इन कॉलों को चार लिंकों के माध्यम से रूट कर सकते हैं?

P5. अनुभाग 1.4 में कार-कारवां सादृश्य की समीक्षा करें । मान लें कि संचरण की गति 100 किमी/घंटा है।

मान लीजिए कि कारवां 150 किलोमीटर की यात्रा करता है, जो एक टोल बूथ के सामने से शुरू होकर दूसरे टोल बूथ से गुज़रता है और तीसरे टोल बूथ के ठीक बाद समाप्त होता है। अंत-से-अंत तक की देरी क्या है?

ख. (क) को दोहराएँ, अब यह मान लें कि कारवां में दस के बजाय आठ कारें हैं।

पी6. यह प्राथमिक समस्या प्रसार विलंब और संचरण विलंब, डेटा नेटवर्किंग में दो केंद्रीय अवधारणाओं का पता लगाने की शुरुआत करती है। दो होस्ट, A और B पर विचार करें, जो एक एकल लिंक से जुड़े हैं

दर आर बीपीएस। मान लीजिए कि दो मेजबान मीटर से अलग हैं , और मान लीजिए कि

लिंक के साथ प्रसार गति s **मीटर/सेकंड है। होस्ट** A **को** L बिट्स आकार का एक पैकेट भेजना है मेजबान बी.

VideoNote

प्रसार विलंब और संचरण विलंब का अन्वेषण

a. प्रसार विलंब को व्यक्त करें, d b. पैकेट का संचरण समय प्रोप , m और s के संदर्भ में .

निर्धारित करें, d ट्रांस , एल और आर के संदर्भ में ।

सी. प्रसंस्करण और कतारबद्ध विलंब की उपेक्षा करते हुए, अंत-से-अंत विलंब के लिए अभिव्यक्ति प्राप्त करें।

मान लीजिए होस्ट A समय पर पैकेट प्रसारित करना शुरू करता है। समय d **पर** पैकेट का अंतिम बिट? टी=0 टी= ट्रांस , जहां है

ई. मान लीजिए d, d से बड़ा है । समय पर ट्रांस टी=डीट्रांस , पैकेट का पहला बिट कहां है?

f. मान लीजिए d, d से कम है । समय पर ट्रांस टी=डीट्रांस , पैकेट का पहला बिट कहां है?

लीजिए s=2.5108 L=120 बिट्स g. मान , और R=56 केबीपीएस , दूरी m ज्ञात करें ताकि d बराबर हो प्रोप

डी ट्रांस ।

P7. इस समस्या में, हम होस्ट A से होस्ट B तक पैकेट-स्विच्ड नेटवर्क (VoIP) पर वास्तविक समय की आवाज़ भेजने पर विचार करते हैं। होस्ट A एनालॉग आवाज़ को तुरंत डिजिटल 64 kbps बिट स्ट्रीम में बदल देता है।

होस्ट A फिर बिट्स को 56-बाइट पैकेट में समूहित करता है। होस्ट A और B के बीच एक लिंक है; यह संचरण दर 2 एमबीपीएस है और इसका प्रसार विलंब 10 एमएसईसी है। जैसे ही होस्ट ए एक संदेश एकत्र करता है पैकेट को होस्ट बी को भेजता है। जैसे ही होस्ट बी को पूरा पैकेट प्राप्त होता है, वह उसे होस्ट बी को भेज देता है। पैकेट के बिट्स को एनालॉग सिग्नल में बदलना। बिट बनने से लेकर एनालॉग सिग्नल बनने तक कितना समय बीतता है? होस्ट A पर मूल एनालॉग सिग्नल) बिट के डिकोड होने तक (होस्ट B पर एनालॉग सिग्नल के भाग के रूप में)?

P8. मान लीजिए कि उपयोगकर्ता 3 एमबीपीएस लिंक साझा करते हैं। साथ ही मान लीजिए कि प्रत्येक उपयोगकर्ता को 150 केबीपीएस की आवश्यकता होती है

पैकेट भेजने की प्रक्रिया में प्रत्येक उपयोगकर्ता केवल 10 प्रतिशत समय ही संचारित करता है। (पैकेट भेजने की प्रक्रिया में प्रत्येक उपयोगकर्ता केवल 10 प्रतिशत समय ही संचारित करता है।)

स्विचिंग बनाम सर्किट स्विचिंग अनुभाग 1.3 **में** ।)

क. जब सर्किट स्विचिंग का उपयोग किया जाता है, तो कितने उपयोगकर्ताओं को समर्थित किया जा सकता है?

b. इस समस्या के शेष भाग के लिए, मान लीजिए पैकेट स्विचिंग का उपयोग किया जाता है। संभावना ज्ञात कीजिए जिसे कोई उपयोगकर्ता प्रेषित कर रहा है।

c. मान लीजिए 120 उपयोगकर्ता हैं। किसी भी समय, ठीक n उपयोगकर्ताओं के **होने की संभावना ज्ञात कीजिए।** एक साथ संचारित हो रहे हैं। (संकेत: द्विपद वितरण का उपयोग करें।)

d. 21 या अधिक उपयोगकर्ताओं द्वारा एक साथ संचार करने की प्रायिकता ज्ञात कीजिए।

पी9. पैकेट स्विचिंग बनाम सर्किट स्विचिंग पर अनुभाग 1.3 में की गई चर्चा पर विचार करें जिसमें एक उदाहरण 1 एमबीपीएस लिंक के साथ दिया गया है। उपयोगकर्ता 100 केबीपीएस की दर से डेटा उत्पन्न कर रहे हैं जब व्यस्त हैं, लेकिन केवल प्रायिकता p=0.1 के साथ डेटा जनरेट करने में व्यस्त हैं मान लीजिए कि 1 एमबीपीएस लिंक है

इसे 1 जीबीपीएस लिंक द्वारा प्रतिस्थापित किया गया।

क. N क्या है , उपयोगकर्ताओं की वह अधिकतम संख्या जो एक साथ समर्थित हो सकती है?

सर्किट स्विचिंग?

b. **अब पैकेट स्विचिंग और** M **उपयोगकर्ताओं** की उपयोगकर्ता जनसंख्या पर विचार करें । एक सूत्र दीजिए (

p, M, N के पदों में) **इस संभावना के लिए कि** N से अधिक उपयोगकर्ता डेटा भेज रहे हैं।

P10. लंबाई L **के एक पैकेट पर विचार करें** जो अंतिम सिस्टम A से शुरू होता है और तीन लिंक से होकर गुजरता है

गंतव्य अंत प्रणाली। ये तीन लिंक दो पैकेट स्विच द्वारा जुड़े हुए हैं। मान लीजिए d, s , और

R **लिंक** i की लम्बाई, प्रसार गति और संचरण दर को दर्शाता है । i=1,2,3

पैकेट स्विच प्रत्येक पैकेट को d **से विलंबित करता है।** d, s (i=1,2,3) के संदर्भ में, कोई कतारबद्ध विलंब न होने की स्थिति में , आर,

, और L, पैकेट के लिए कुल अंत-से-अंत देरी क्या है? मान लीजिए अब पैकेट है

1,500 बाइट्स, तीनों लिंक पर प्रसार गति है 2.5108मी/सेकंड, सभी की संचरण दरें

तीन लिंक 2 एमबीपीएस हैं, पैकेट स्विच प्रसंस्करण देरी 3 एमएसईसी है, पहले लिंक की लंबाई है

5,000 किमी, दूसरे लिंक की लंबाई 4,000 किमी तथा अंतिम लिंक की लंबाई 1,000 किमी है।

इन मानों के लिए, अंत-से-अंत विलंब क्या है?

P11. उपरोक्त समस्या में, मान लीजिए और । इसके अलावा मान लीजिए पैकेट आर1=आर2=आर3=आर डीप्रोक=0

स्विच पैकेट को स्टोर-एंड-फॉरवर्ड नहीं करता है, बल्कि प्राप्त प्रत्येक बिट को तुरंत प्रसारित करता है

पूरे पैकेट के आने का इंतज़ार करने से पहले। एंड-टू-एंड देरी क्या है?

P12. पैकेट स्विच एक पैकेट प्राप्त करता है और पैकेट को जिस आउटबाउंड लिंक पर ले जाना है उसे निर्धारित करता है

पैकेट जब आता है, तो एक अन्य पैकेट आधा भेजा जा चुका होता है।

इस आउटबाउंड लिंक पर चार पैकेट भेजे जा रहे हैं और चार अन्य पैकेट भेजे जाने की प्रतीक्षा कर रहे हैं।

आगमन के क्रम में प्रेषित किए जाते हैं। मान लीजिए कि सभी पैकेट 1,500 बाइट्स के हैं और लिंक दर 2 है

एमबीपीएस। पैकेट के लिए कतारबद्ध विलंब क्या है? अधिक सामान्यतः, कतारबद्ध विलंब क्या है जब

सभी पैकेट की लंबाई L है, संचरण दर R है, वर्तमान में प्रेषित होने वाले x **बिट्स**

पैकेट प्रेषित हो चुके हैं, और n पैकेट पहले से ही कतार में हैं?

पी13.

a. मान लीजिए N पैकेट एक साथ किसी लिंक पर पहुंचते हैं, जहां वर्तमान में कोई पैकेट नहीं है

प्रेषित या कतारबद्ध किया जा रहा है। प्रत्येक पैकेट की लंबाई L **है** और लिंक में संचरण होता है

दर **R.** N **पैकेटों** के लिए औसत कतार विलंब क्या है ?

अब मान लीजिए कि **प्रत्येक** LN/R सेकंड में N ऐसे पैकेट लिंक पर आते हैं ।

एक पैकेट की औसत कतारबद्ध देरी?

P14. राउटर बफर में कतारबद्ध विलंब पर विचार करें। मान लें कि I ट्रैफ़िक तीव्रता को दर्शाता है; यानी, मान लीजिए कि कतारबद्ध विलंब I<1 के लिए IL/R(1–I) का I=ला/आर .

रूप लेता है .

क. कुल विलंब के लिए एक सूत्र प्रदान करें, अर्थात कतारबद्ध विलंब + संचरण विलंब

देरी।

ख. कुल विलंब को L/R के फलन के रूप में आलेखित करें।

P15. मान लें कि a पैकेट/सेकंड में लिंक पर पहुंचने वाले पैकेट की दर को दर्शाता है, और µ लिंक की गति को दर्शाता है।

पैकेट/सेकंड में संचरण दर। कुल विलंब (यानी कतारबद्ध विलंब) के सूत्र के आधार पर

पिछली समस्या में प्राप्त कुल विलंब (+ संचरण विलंब) का उपयोग करके, कुल विलंब के लिए एक सूत्र निकालें

a और μ के संदर्भ में .

P16. आउटबाउंड लिंक से पहले राउटर बफर पर विचार करें। इस समस्या में, आप लिटिल के सूत्र का उपयोग करेंगे, जो कतार सिद्धांत से एक प्रसिद्ध सूत्र है। N को बफर में

पैकेट की औसत संख्या और प्रेषित किए जा रहे पैकेट को दर्शाएँ। a को लिंक पर आने वाले पैकेट की दर को दर्शाएँ।

मान लीजिए d एक पैकेट द्वारा अनुभव की गई औसत कुल देरी (यानी, कतारबद्ध देरी और ट्रांसमिशन देरी) को दर्शाता है। लिटिल का सूत्र है। मान लीजिए कि औसतन, बफर में

$N=a \cdot d$ है

10 पैकेट, और औसत पैकेट कतारबद्ध विलंब 10 msec है। लिंक की संचरण दर है

100 पैकेट/सेकंड। लिटिल के सूत्र का उपयोग करते हुए, औसत पैकेट आगमन दर क्या है, यह मानते हुए कि कोई पैकेट हानि नहीं है?

पी17.

a. विषम प्रसंस्करण दरों के लिए अनुभाग 1.4.3 में समीकरण 1.2 को सामान्यीकृत करें ,
संचरण दर और प्रसार विलंब.

ख. **(ए) को दोहराएँ, लेकिन अब यह भी मान लें कि** प्रत्येक पर औसत कतारबद्धता विलंब d **है** कतार
नोड.

P18. एक ही महाद्वीप पर स्रोत और गंतव्य के बीच तीन स्थानों पर ट्रेसरूट का संचालन करें

दिन के अलग-अलग समय पर।

VideoNote

नेटवर्क पथों की खोज करने और नेटवर्क विलंब को मापने के लिए ट्रेसरूट का उपयोग करना

a. तीनों में से प्रत्येक पर राउंड-ट्रिप देरी का औसत और मानक विचलन ज्ञात करें
घंटे।

ख. तीनों घंटों में से प्रत्येक पर पथ में राउटरों की संख्या ज्ञात करें। क्या किसी भी घंटे के दौरान पथ बदल गया?

सी. ट्रेसरूट पैकेट स्रोत से गंतव्य तक कितने ISP नेटवर्क से होकर गुज़रते हैं, इसकी पहचान करने की कोशिश करें। समान नाम और/या समान IP पते वाले राउटर को अलग-अलग
होना चाहिए।

एक ही ISP के हिस्से के रूप में माना जाता है। आपके प्रयोगों में, क्या सबसे बड़ी देरी आसन्न ISP के बीच पियरिंग इंटरफेस पर होती है?

d. विभिन्न महाद्वीपों पर स्रोत और गंतव्य के लिए उपरोक्त को दोहराएं।
अंतर-महाद्वीपीय और अंतर-महाद्वीपीय परिणाम।

पी19.

a. www.traceroute.org साइट पर जाएँ और दो अलग-अलग शहरों से ट्रेसरूट का प्रदर्शन करें
फ्रांस से संयुक्त राज्य अमेरिका में एक ही गंतव्य होस्ट तक। कितने लिंक एक जैसे हैं

दो ट्रेसरूट में क्या अंतर है? क्या ट्रान्साटलांटिक लिंक एक ही है?

(ख) (क) को दोहराएँ लेकिन इस बार फ्रांस का एक शहर और जर्मनी का दूसरा शहर चुनें।

c. संयुक्त राज्य अमेरिका में एक शहर चुनें, और दो होस्टों के लिए ट्रेसरूट निष्पादित करें, प्रत्येक एक अलग स्थान पर चीन के शहर में स्थित है। दोनों ट्रेसरूट में कितने लिंक कॉमन हैं? क्या दोनों ट्रेसरूट एक ही हैं? क्या ट्रेसरूट चीन पहुंचने से पहले ही अलग हो जाते हैं?

P20. चित्र 1.20(b) के अनुरूप थ्रूपुट उदाहरण पर विचार करें । अब मान लीजिए कि 10 के बजाय M **क्लाइंट-सर्वर जोड़े** हैं । R लिंक, क्लाइंट लिंक और नेटवर्क लिंक को निरूपित करें। मान एस, आर$_{सी}$, और R सर्वर की दरों के लिए लें कि अन्य सभी लिंक में प्रचुर क्षमता है और वहाँ नेटवर्क में M क्लाइंट-सर्वर जोड़ों द्वारा उत्पन्न ट्रैफ़िक के अलावा कोई अन्य ट्रैफ़िक नहीं है। R के संदर्भ में थ्रूपुट के लिए एक सामान्य अभिव्यक्ति एस, आर$_{सी}$, आर, और एम.

P21. चित्र 1.19(b) **पर विचार करें। अब मान लीजिए कि** सर्वर और सर्वर के बीच M **पथ** हैं। क्लाइंट। कोई भी दो पथ कोई लिंक साझा नहीं करते। पथ में ट्रांसमिशन दरों के साथ N लिंक (के=1...एम) **शामिल हैं** आर1के,आर2के,...,आरएनके यदि सर्वर क्लाइंट को डेटा भेजने के लिए केवल एक पथ का उपयोग कर सकता है, तो क्या होगा? सर्वर अधिकतम कितना थ्रूपुट प्राप्त कर सकता है? यदि सर्वर डेटा भेजने के लिए **सभी** M **पथों का उपयोग कर सकता है,** सर्वर द्वारा प्राप्त की जा सकने वाली अधिकतम थ्रूपुट क्या है?

P22. चित्र 1.19(b) **पर विचार करें** । मान लीजिए कि सर्वर और क्लाइंट के बीच प्रत्येक लिंक में एक है पैकेट हानि संभावना p, और इन लिंक के लिए पैकेट हानि संभावनाएँ स्वतंत्र हैं। क्या संभावना है कि एक पैकेट (सर्वर द्वारा भेजा गया) रिसीवर द्वारा सफलतापूर्वक प्राप्त किया गया है? यदि यदि सर्वर से क्लाइंट तक जाने के मार्ग में कोई पैकेट खो जाता है, तो सर्वर पैकेट को पुनः प्रेषित कर देगा। औसतन, क्लाइंट को भेजने के लिए सर्वर कितनी बार पैकेट को पुनः प्रेषित करेगा? पैकेट सफलतापूर्वक प्राप्त हुआ?

P23. चित्र 1.19(a) **पर विचार करें** । मान लें कि हम पथ के साथ-साथ अड़चन लिंक को जानते हैं सर्वर से क्लाइंट तक पहुँचने वाला पहला लिंक R बिट्स/सेकंड की दर से है। मान लीजिए कि हम पैकेटों की एक जोड़ी वापस भेजते हैं सर्वर से क्लाइंट तक वापस जाने के लिए, और इस पथ पर कोई अन्य ट्रैफ़िक नहीं है। मान लें कि प्रत्येक पैकेट का आकार L बिट्स है, और दोनों लिंक में समान प्रसार विलंब d **है** । प्रोप

गंतव्य पर पैकेट के पहुंचने में कितना समय लगता है? यानी, कितना समय बीतता है पहले पैकेट के अंतिम बिट के आने से लेकर दूसरे पैकेट के अंतिम बिट तक आता है?

अब मान लीजिए कि दूसरी कड़ी अड़चन कड़ी है (यानी,)। क्या यह संभव है कि आरसी<रु दूसरे लिंक की इनपुट कतार में दूसरा पैकेट कतारबद्ध है? समझाइए। अब मान लीजिए सर्वर पहला पैकेट भेजने के T **सेकंड बाद दूसरा पैकेट भेजता है।** दूसरे लिंक से पहले कोई कतार न लगे, यह सुनिश्चित करने के लिए T कितना बड़ा होना चाहिए ? **समझाइए।**

P24. मान लीजिए कि आप बोस्टन से लॉस एंजिल्स तक 40 टेराबाइट डेटा तत्काल पहुंचाना चाहते हैं। आपके पास डेटा ट्रांसफर के लिए 100 एमबीपीएस समर्पित लिंक उपलब्ध है। क्या आप इसे ट्रांसमिट करना पसंद करेंगे? इस लिंक के माध्यम से डेटा प्राप्त करें या इसके बजाय FedEx ओवर-नाइट डिलीवरी का उपयोग करें? स्पष्ट करें।

P25. मान लीजिए दो होस्ट, A और B, 20,000 किलोमीटर की दूरी पर हैं और एक दूसरे से जुड़े हुए हैं एमबीपीएस का सीधा आर=2 2.5 108 लिंक। मान लीजिए लिंक पर प्रसार गति मीटर/सेकंड है।

a. बैंडविड्थ-विलंब उत्पाद की गणना करें, R dप्रॉप .

b. होस्ट A से होस्ट B तक 800,000 बिट्स की फ़ाइल भेजने पर विचार करें। मान लीजिए कि फ़ाइल को एक बड़े संदेश के रूप में लगातार भेजा जाता है। किसी भी समय लिंक में अधिकतम कितने बिट्स होंगे?

c. बैंडविड्थ-देरी उत्पाद की व्याख्या प्रदान करें। d. लिंक में बिट की चौड़ाई (मीटर में) क्या है? क्या यह फुटबॉल के मैदान से अधिक लंबी है? e . प्रसार गति s के संदर्भ में बिट की चौड़ाई के लिए एक सामान्य अभिव्यक्ति प्राप्त करें ,

संचरण दर R, तथा लिंक की लम्बाई m.

P26. समस्या P25 का संदर्भ लेते हुए, मान लीजिए कि हम R को संशोधित कर सकते हैं। R के किस मान के लिए बिट की चौड़ाई लिंक की लंबाई के बराबर होगी?

P27. समस्या P25 पर विचार करें लेकिन अब Gbps के लिंक के साथ। R=1

a. बैंडविड्थ-विलंब उत्पाद की गणना करें, b. होस्ट A से होस्ट B तक R dprop .

800,000 बिट्स की फ़ाइल भेजने पर विचार करें। मान लीजिए फ़ाइल भेजी जाती है लगातार एक बड़े संदेश के रूप में। किसी भी समय लिंक में अधिकतम कितने बिट्स होंगे?

ग. लिंक में बिट की चौड़ाई (मीटर में) क्या है?

प्र.28. समस्या प्र.25 को पुनः देखें।

ए. फ़ाइल को भेजने में कितना समय लगता है, यह मानते हुए कि इसे लगातार भेजा जाता है? बी. मान लीजिए कि अब फ़ाइल को 20 पैकेट में विभाजित किया गया है, जिसमें प्रत्येक पैकेट में 40,000 बिट्स हैं। मान लीजिए कि प्रत्येक पैकेट को रिसीवर द्वारा स्वीकार किया जाता है और एक पावती पैकेट का संचरण समय नगण्य है। अंत में, मान लें कि प्रेषक नहीं कर सकता

पैकेट को तब तक भेजें जब तक कि पिछले पैकेट को स्वीकार न कर लिया जाए। भेजने में कितना समय लगता है फ़ाइल?

(ग) (क) और (ख) के परिणामों की तुलना करें।

P29. मान लीजिए कि भूस्थिर उपग्रह और पृथ्वी पर उसके बेस स्टेशन के बीच 10 एमबीपीएस माइक्रोवेव लिंक है। हर मिनट उपग्रह एक डिजिटल फोटो लेता है और उसे बेस स्टेशन पर भेजता है।

मान लीजिए प्रसार गति मीटर/सेकेंड है। 2.4 108

क. लिंक का प्रसार विलंब क्या है?

बैंडविड्थ-देरी उत्पाद क्या है? c. मान लें कि x फोटो के आकार को दर्शाता है। माइक्रोवेव लिंक के R dprop b.

लिए x का न्यूनतम मान क्या है लगातार संचारित किया जाना है?

P30. धारा 1.5 में लेयरिंग पर हमारी चर्चा में एयरलाइन यात्रा सादृश्य पर विचार करें, जिसमें प्रोटोकॉल डेटा इकाइयों में हेडर जोड़े जाते हैं क्योंकि वे , और यह प्रोटोकॉल स्टैक के नीचे प्रवाहित होते हैं। क्या हेडर सूचना की कोई समतुल्य धारणा है जिसे यात्रियों और सामान में जोड़ा जाता है क्योंकि वे एयरलाइन प्रोटोकॉल स्टैक के नीचे जाते हैं?

P31. आधुनिक पैकेट-स्विच्ड नेटवर्क में, जिसमें इंटरनेट भी शामिल है, स्रोत होस्ट लंबे, एप्लिकेशन-लेयर संदेशों (उदाहरण के लिए, एक छवि या एक संगीत फ़ाइल) को छोटे पैकेटों में विभाजित करता है

और पैकेट को नेटवर्क में भेजता है। रिसीवर फिर पैकेट को वापस नेटवर्क में जोड़ता है

मूल संदेश। हम इस प्रक्रिया को संदेश विभाजन कहते हैं। चित्र 1.27 संदेश विभाजन के साथ और बिना संदेश के अंत-से-अंत परिवहन को दर्शाता है।

चित्र 1.27

में 8□106 बिट्स लम्बा संदेश जो स्रोत से गंतव्य तक भेजा जाना है ।

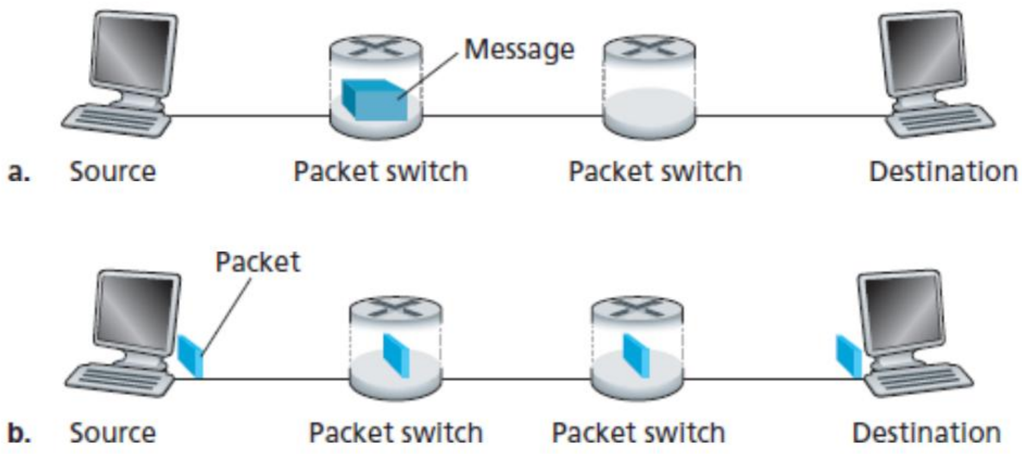
मान लीजिए कि चित्र में प्रत्येक लिंक 2 एमबीपीएस है। प्रसार, कतार और प्रसंस्करण देरी को अनदेखा करें।

a. स्रोत से गंतव्य तक संदेश के बिना संदेश **भेजने पर विचार करें** विभाजन। संदेश को स्रोत होस्ट से पहले पैकेट स्विच तक ले जाने में कितना समय लगता है? यह ध्यान में रखते हुए कि प्रत्येक स्विच स्टोर-एंड-फॉरवर्ड पैकेट स्विचिंग का उपयोग करता है, संदेश को स्रोत होस्ट से गंतव्य तक ले जाने में कुल कितना समय लगता है

मेज़बान?

अब मान लीजिए कि संदेश को 800 पैकेट में विभाजित किया गया है, जिसमें प्रत्येक पैकेट 10,000 बिट लंबा है। पहले पैकेट को सोर्स होस्ट से पहले स्विच तक ले जाने में कितना समय लगता है? जब पहला पैकेट पहले स्विच से दूसरे स्विच पर भेजा जा रहा है, तो दूसरा पैकेट सोर्स होस्ट से पहले स्विच पर भेजा जा रहा है। दूसरे पैकेट को पहले स्विच पर पूरी तरह से कब प्राप्त किया जाएगा?

c. फ़ाइल को स्रोत होस्ट से गंतव्य होस्ट तक ले जाने में कितना समय लगता है? संदेश विभाजन का उपयोग किया जाता है? इस परिणाम की तुलना भाग (ए) और (ए) में अपने उत्तर से करें टिप्पणी।



चित्र 1.27 अंत-से-अंत संदेश परिवहन: (ए) संदेश विभाजन के बिना; (बी) संदेश विभाजन के साथ

d. देरी को कम करने के अलावा, संदेश विभाजन का उपयोग करने के क्या कारण हैं? e. संदेश विभाजन की कमियों पर चर्चा करें।

P32. पुस्तक की वेब साइट पर संदेश विभाजन एप्लेट के साथ प्रयोग करें। क्या एप्लेट में देरी पिछली समस्या में देरी के अनुरूप है? लिंक प्रसार देरी पैकेट स्विचिंग (संदेश विभाजन के साथ) और संदेश स्विचिंग के लिए समग्र अंत-से-अंत देरी को कैसे प्रभावित करती है?

P33. **होस्ट A से होस्ट B तक** F **बिट्स** की एक बड़ी फ़ाइल भेजने पर विचार करें। A और B के बीच तीन लिंक (और दो स्विच) हैं, और लिंक अनकॉन्जस्टेड हैं (यानी, कोई कतारबद्ध देरी नहीं)। होस्ट A

फ़ाइल को S बिट्स के खंडों में विभाजित करता है और प्रत्येक खंड में 80 बिट्स हेडर जोड़ता है, L=80 + S **बिट्स के पैकेट बनाता है। प्रत्येक लिंक में** R bps की संचरण दर होती है LS का वह मान ज्ञात करें जो होस्ट A से होस्ट B तक फ़ाइल ले जाने में होने वाली देरी को न्यूनतम करता है। प्रसार विलंब पर ध्यान न दें।

P34. Skype एक ऐसी सेवा प्रदान करता है जो आपको PC से किसी साधारण फ़ोन पर फ़ोन कॉल करने की अनुमति देता है। इसका मतलब है कि वॉयस कॉल को इंटरनेट और टेलीफ़ोन नेटवर्क दोनों से होकर गुज़रना चाहिए। चर्चा करें कि यह कैसे किया जा सकता है।

वायरशार्क लैब

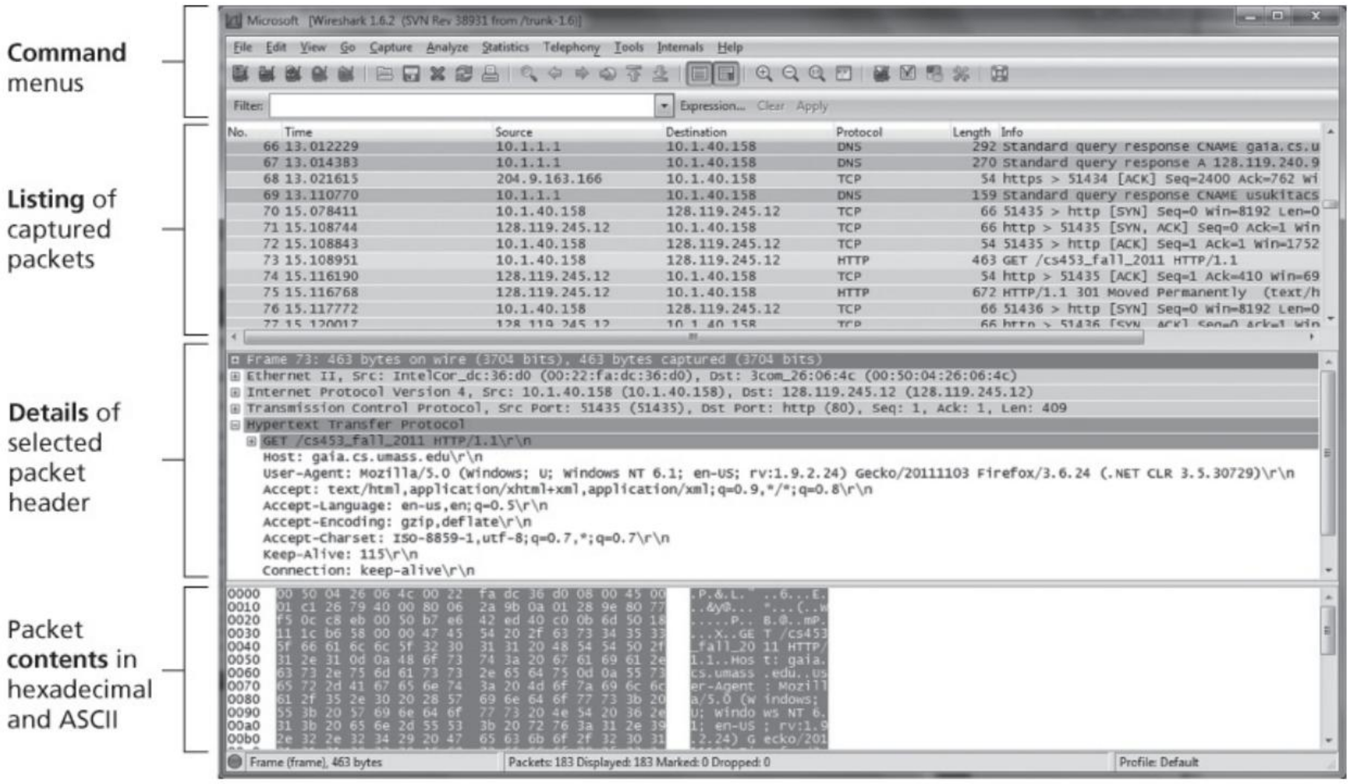
> "मुझे बताओ और मैं भूल जाऊंगा। मुझे दिखाओ और मैं याद रखूंगा। मुझे शामिल करो और मैं समझ जाऊंगा।"
>
> चीनी कहावत

नेटवर्क प्रोटोकॉल की समझ को अक्सर उन्हें क्रियान्वित होते देखकर और उनके साथ खेलकर बहुत गहरा किया जा सकता है - दो प्रोटोकॉल इकाइयों के बीच आदान-प्रदान किए गए संदेशों के अनुक्रम का अवलोकन करके, प्रोटोकॉल संचालन के विवरण में गहराई से जाना, प्रोटोकॉल को कुछ क्रियाएं करने के लिए प्रेरित करना, और इन क्रियाओं और उनके परिणामों का अवलोकन करना। यह सिम्युलेटेड परिदृश्यों में या इंटरनेट जैसे वास्तविक नेटवर्क वातावरण में किया जा सकता है। पाठ्यपुस्तक वेब साइट पर जावा एप्लेट पहला दृष्टिकोण अपनाते हैं। वायरशार्क प्रयोगशालाओं में, हम दूसरा दृष्टिकोण अपनाएंगे। आप अपने डेस्क पर, घर पर या प्रयोगशाला में कंप्यूटर का उपयोग करके विभिन्न परिदृश्यों में नेटवर्क एप्लिकेशन चलाएंगे।

आप अपने कंप्यूटर में नेटवर्क प्रोटोकॉल का निरीक्षण करेंगे, इंटरनेट पर कहीं और क्रियान्वित प्रोटोकॉल इकाइयों के साथ बातचीत और संदेशों का आदान-प्रदान करेंगे। इस प्रकार, आप और आपका कंप्यूटर इन लाइव प्रयोगशालाओं का एक अभिन्न अंग होंगे। आप निरीक्षण करेंगे - और आप करके सीखेंगे।

निष्पादित प्रोटोकॉल इकाइयों के बीच आदान-प्रदान किए गए संदेशों को देखने के लिए मूल उपकरण को पैकेट स्निफ़र कहा जाता है। जैसा कि नाम से पता चलता है, एक पैकेट स्निफ़र आपके कंप्यूटर से भेजे और प्राप्त किए जा रहे संदेशों को निष्क्रिय रूप से कॉपी (सूँघता) है; यह इन कैप्चर किए गए संदेशों के विभिन्न प्रोटोकॉल फ़ील्ड की सामग्री भी प्रदर्शित करता है।

वायरशार्क पैकेट स्निफर को चित्र 1.28 में दिखाया गया है। वायरशार्क एक निःशुल्क पैकेट स्निफर है जो विंडोज, लिनक्स/यूनिक्स और मैक कंप्यूटर पर चलता है।



चित्र 1.28 वायरशार्क स्क्रीनशॉट (वायरशार्क स्क्रीनशॉट द्वारा पुनर्मुद्रित)

वायरशार्क फाउंडेशन की अनुमति से)

पूरी पाठ्यपुस्तक में, आपको Wireshark प्रयोगशालाएँ मिलेंगी जो आपको अध्याय में अध्ययन किए गए कई प्रोटोकॉल का पता लगाने की अनुमति देती हैं। इस पहली Wireshark प्रयोगशाला में, आप Wireshark की एक प्रति प्राप्त करेंगे और उसे स्थापित करेंगे, एक वेब साइट तक पहुँचेंगे, और अपने वेब ब्राउज़र और वेब सर्वर के बीच आदान-प्रदान किए जा रहे प्रोटोकॉल संदेशों को कैप्चर और जाँचेंगे।

आप इस पहली वायरशार्क प्रयोगशाला के बारे में पूरी जानकारी पा सकते हैं (इसमें निर्देश भी शामिल हैं कि कैसे Wireshark प्राप्त करें और उसे स्थापित करें) वेब साइट http://www.pearsonhighered.com/cs- पर संसाधन/।

एक साक्षात्कार...

लियोनार्ड क्लेनरॉक

लियोनार्ड क्लेनरॉक कैलिफोर्निया विश्वविद्यालय, लॉस एंजिल्स में कंप्यूटर विज्ञान के प्रोफेसर हैं। 1969 में, UCLA में उनका कंप्यूटर इंटरनेट का पहला नोड बन गया। 1961 में पैकेट-स्विचिंग सिद्धांतों का उनका निर्माण इंटरनेट के पीछे की तकनीक बन गया। उन्होंने सिटी कॉलेज ऑफ़ न्यूयॉर्क (CCNY) से BEE और MIT से इलेक्ट्रिकल इंजीनियरिंग में मास्टर और पीएचडी की उपाधि प्राप्त की।



आपने नेटवर्किंग/इंटरनेट प्रौद्योगिकी में विशेषज्ञता हासिल करने का निर्णय क्यों लिया?

1959 में एमआईटी में पीएचडी छात्र के रूप में, मैंने चारों ओर देखा और पाया कि मेरे अधिकांश सहपाठी सूचना सिद्धांत और कोडिंग सिद्धांत के क्षेत्र में शोध कर रहे थे। एमआईटी में, महान थे शोधकर्ता, क्लाउड शैनन, जिन्होंने इन क्षेत्रों को लॉन्च किया था और अधिकांश समस्याओं को हल किया था महत्वपूर्ण समस्याओं का समाधान पहले से ही था। जो शोध समस्याएँ बची थीं, वे कठिन और कम महत्वपूर्ण थीं। इसलिए मैंने एक नए क्षेत्र में काम शुरू करने का फैसला किया, जिसके बारे में अभी तक किसी और ने नहीं सोचा था। याद कीजिए कि एम.आई.टी. में मैं बहुत सारे कंप्यूटरों से घिरा हुआ था, और यह बात मेरे लिए जल्द ही स्पष्ट हो गई थी कि इन मशीनों को एक दूसरे के साथ संवाद करने की आवश्यकता होगी। उस समय, कोई प्रभावी तरीका नहीं था उनके लिए ऐसा करने का कोई तरीका नहीं है, इसलिए मैंने ऐसी तकनीक विकसित करने का निर्णय लिया जो कुशल और विश्वसनीय डेटा नेटवर्क बनाए जाएंगे।

कंप्यूटर उद्योग में आपकी पहली नौकरी क्या थी? इसमें क्या-क्या शामिल था?

मैं 1951 से 1957 तक CCNY में इलेक्ट्रिकल इंजीनियरिंग में स्नातक की डिग्री के लिए शाम के सत्र में गया। दिन के दौरान, मैंने पहले एक तकनीशियन के रूप में और फिर फोटोबेल नामक एक छोटी, औद्योगिक इलेक्ट्रॉनिक्स फर्म में एक इंजीनियर के रूप में काम किया। वहाँ रहते हुए, मैंने उनके उत्पाद लाइन में डिजिटल तकनीक पेश की। अनिवार्य रूप से, हम कुछ की उपस्थिति का पता लगाने के लिए फोटोइलेक्ट्रिक उपकरणों का उपयोग कर रहे थे

वस्तुओं (बक्से, लोग, आदि) और एक सर्किट का उपयोग जिसे तब बिस्टेबल मल्टीवाइब्रेटर **के रूप में जाना जाता था ,** ठीक उसी प्रकार की तकनीक थी जिसकी हमें पता लगाने के इस क्षेत्र में डिजिटल प्रसंस्करण लाने के लिए आवश्यकता थी। ये सर्किट कंप्यूटर के लिए बिल्डिंग ब्लॉक होते हैं, और इन्हें के रूप में जाना जाता है आज की भाषा में इसे फ्लिप-फ्लॉप या स्विच कहा जाता है ।

जब आपने पहला होस्ट-टू-होस्ट संदेश (यूसीएलए से) भेजा तो आपके मन में क्या चल रहा था?

स्टैनफोर्ड रिसर्च इंस्टीट्यूट)?

सच कहें तो हमें उस आयोजन के महत्व का कोई अंदाजा नहीं था। हमने कोई विशेष तैयारी नहीं की थी ऐतिहासिक महत्व का संदेश, जैसा कि अतीत के कई आविष्कारकों ने किया था (सैमुअल मोर्स ने "व्हाट भगवान ने बनाया है।" या अलेक्जेंडर ग्राहम बेल के साथ "वॉटसन, यहाँ आओ! मैं तुम्हें चाहता हूँ।" या नील एम्स्ट्रोंग के साथ "यह एक आदमी के लिए एक छोटा कदम है, मानव जाति के लिए एक बड़ी छलांग है।") वे लोग थे

होशियार! वे मीडिया और जनसंपर्क को समझते थे। हमें बस SRI कंप्यूटर में लॉग इन करना था। इसलिए हमने "L" टाइप किया, जो सही तरीके से प्राप्त हुआ, हमने "o" टाइप किया जो प्राप्त हुआ, और फिर हमने "g" टाइप किया जिससे SRI होस्ट कंप्यूटर क्रैश हो गया! तो, यह पता चला कि हमारा संदेश अब तक का सबसे छोटा और शायद सबसे भविष्यसूचक संदेश था, अर्थात

"लो!" जैसे कि "देखो और देखो!"

उस वर्ष की शुरुआत में, यूसीएलए प्रेस विज्ञप्ति में मुझे यह कहते हुए उद्धृत किया गया था कि एक बार नेटवर्क चालू हो जाने के बाद, हमारे घरों और कार्यालयों से कंप्यूटर उपयोगिताओं तक पहुँच प्राप्त करना उतना ही आसान होगा जितना कि हम बिजली और टेलीफोन कनेक्टिविटी तक पहुँच प्राप्त करते हैं। इसलिए उस समय मेरा दृष्टिकोण यह था कि इंटरनेट सर्वव्यापी होगा, हमेशा चालू रहेगा, हमेशा उपलब्ध रहेगा, कोई भी व्यक्ति किसी भी डिवाइस से किसी भी स्थान से जुड़ सकता है, और यह अदृश्य होगा। हालाँकि, मैंने कभी नहीं सोचा था कि मेरी 99 वर्षीय माँ इंटरनेट का उपयोग करेंगी - और वास्तव में उन्होंने किया!

नेटवर्किंग के भविष्य के लिए आपका दृष्टिकोण क्या है?

विज़न का आसान हिस्सा बुनियादी ढांचे की भविष्यवाणी करना है। मुझे लगता है कि हम खानाबदोश कंप्यूटिंग, मोबाइल डिवाइस और स्मार्ट स्पेस की काफी तैनाती देखेंगे। वास्तव में, हल्के, सस्ते, उच्च प्रदर्शन वाले, पोर्टेबल कंप्यूटिंग और संचार उपकरणों (साथ ही इंटरनेट की सर्वव्यापकता) की उपलब्धता ने हमें खानाबदोश बनने में सक्षम बनाया है।

खानाबदोश कंप्यूटिंग उस तकनीक को संदर्भित करती है जो एक स्थान से दूसरे स्थान पर यात्रा करने वाले अंतिम उपयोगकर्ताओं को पारदर्शी तरीके से इंटरनेट सेवाओं तक पहुँच प्राप्त करने में सक्षम बनाती है, चाहे वे कहीं भी यात्रा करें और चाहे वे कोई भी उपकरण ले जाएँ या जिस तक पहुँच प्राप्त करें। विज़न का सबसे कठिन हिस्सा उन अनुप्रयोगों और सेवाओं की भविष्यवाणी करना है, जिन्होंने हमें लगातार नाटकीय तरीकों से आश्चर्यचकित किया है (ई-मेल, खोज तकनीक, वर्ल्ड वाइड वेब, ब्लॉग, सोशल नेटवर्क, उपयोगकर्ता पीढ़ी, और संगीत, फ़ोटो और वीडियो साझा करना, आदि)। हम अपने हाथ में पकड़े जाने वाले उपकरणों पर वितरित आश्चर्यजनक और अभिनव मोबाइल अनुप्रयोगों की एक नई श्रेणी के कगार पर हैं।

अगला कदम हमें साइबरस्पेस की अधोलोक से स्मार्ट स्पेस की भौतिक दुनिया में ले जाने में सक्षम बनाएगा। हमारे वातावरण (डेस्क, दीवारें, वाहन, घड़ियाँ, बेल्ट, इत्यादि)
एक्ट्यूएटर्स, सेंसर्स, लॉजिक, प्रोसेसिंग, स्टोरेज, कैमरा आदि के माध्यम से प्रौद्योगिकी के साथ जीवंत हो उठें।
माइक्रोफोन, स्पीकर, डिस्प्ले और संचार। यह एम्बेडेड तकनीक हमारे
हम जो आईपी सेवाएँ चाहते हैं, उन्हें प्रदान करने के लिए वातावरण। जब मैं किसी कमरे में प्रवेश करता हूँ, तो कमरा जान जाता है कि मैं
मैं अपने परिवेश के साथ स्वाभाविक रूप से संवाद करने में सक्षम हो जाऊंगा, जैसे कि बोली जाने वाली अंग्रेजी में; मेरी
अनुरोधों से उत्तर उत्पन्न होंगे जो दीवार के डिस्प्ले से, मेरे चश्मे के माध्यम से, भाषण, होलोग्राम आदि के रूप में वेब पेजों को मेरे सामने प्रस्तुत करेंगे।

थोड़ा और आगे देखने पर, मुझे एक ऐसा नेटवर्किंग भविष्य दिखाई देता है जिसमें निम्नलिखित अतिरिक्त मुख्य घटक शामिल हैं। मैं नेटवर्क में तैनात बुद्धिमान सॉफ़्टवेयर एजेंट देखता हूँ जिनका कार्य डेटा को माइन करना, उस डेटा पर कार्य करना, रुझानों का निरीक्षण करना और गतिशील और अनुकूली रूप से कार्य करना है। मैं काफी अधिक नेटवर्क ट्रैफ़िक देखता हूँ जो मनुष्यों द्वारा नहीं, बल्कि इन एम्बेडेड डिवाइस और इन बुद्धिमान सॉफ़्टवेयर एजेंटों द्वारा उत्पन्न होता है। मैं इस विशाल, तेज़ नेटवर्क को नियंत्रित करने वाले स्व-संगठित सिस्टम के बड़े संग्रह को देखता हूँ। मैं बड़ी मात्रा में सूचना को फ्लैश होते हुए देखता हूँ

इस नेटवर्क पर तुरंत ही इस सूचना को बहुत अधिक प्रसंस्करण और फ़िल्टरिंग से गुजरना होगा। इंटरनेट अनिवार्य रूप से एक व्यापक वैश्विक तंत्रिका तंत्र होगा। मैं इन सभी चीजों और अधिक को देखता हूं क्योंकि हम इक्कीसवीं सदी में तेजी से आगे बढ़ रहे हैं।

पेशेवर तौर पर किन लोगों ने आपको प्रेरित किया है?

अब तक, यह एमआईटी के क्लाउड शैनन थे, जो एक शानदार शोधकर्ता थे, जिनके पास अपने गणितीय विचारों को भौतिक दुनिया से अत्यंत सहज तरीकों से जोड़ने की क्षमता थी। वे मेरे पीएचडी थीसिस पर थे।
समिति।

क्या आपके पास नेटवर्किंग/इंटरनेट क्षेत्र में प्रवेश करने वाले छात्रों के लिए कोई सलाह है?

इंटरनेट और इससे जुड़ी सभी चीजें एक विशाल नई सीमा है, जो आश्चर्यजनक चुनौतियों से भरी हुई है। महान नवाचार के लिए जगह है। आज की तकनीक से विवश न हों। आगे बढ़ें और कल्पना करें कि क्या हो सकता है और फिर उसे साकार करें।

अध्याय 2 अनुप्रयोग परत

नेटवर्क अनुप्रयोग, कंप्यूटर नेटवर्क के अस्तित्व का कारण **हैं - यदि हम किसी उपयोगी अनुप्रयोग की कल्पना नहीं कर सकते, तो उन्हें समर्थन देने के लिए नेटवर्किंग अवसंरचना और प्रोटोकॉल की कोई आवश्यकता नहीं होगी।**

इंटरनेट की शुरुआत के बाद से, कई उपयोगी और मनोरंजक अनुप्रयोग बनाए गए हैं।

ये अनुप्रयोग इंटरनेट की सफलता के पीछे प्रेरक शक्ति रहे हैं, तथा घरों, स्कूलों, सरकारों और व्यवसायों में लोगों को इंटरनेट को अपने दैनिक जीवन का अभिन्न अंग बनाने के लिए प्रेरित करते रहे हैं।

गतिविधियाँ।

इंटरनेट अनुप्रयोगों में क्लासिक टेक्स्ट-आधारित अनुप्रयोग शामिल हैं जो 1970 के दशक में लोकप्रिय हुए और

1980 का दशक: टेक्स्ट ई-मेल, कंप्यूटर तक रिमोट एक्सेस, फ़ाइल ट्रांसफ़र और न्यूज़ग्रुप। इनमें 1990 के दशक के मध्य का किलर एप्लीकेशन, वर्ल्ड वाइड वेब शामिल है , जिसमें वेब सर्फिंग, सर्च और इलेक्ट्रॉनिक कॉमर्स शामिल हैं। इनमें इंस्टेंट मैसेजिंग और P2P फ़ाइल शेयरिंग शामिल हैं, जो सहस्राब्दी के अंत में पेश किए गए दो किलर एप्लीकेशन हैं। नई सहस्राब्दी में, नए और बेहद आकर्षक एप्लीकेशन उभर कर सामने आते रहते हैं, जिसमें वॉयस ओवर IP और वीडियो कॉन्फ्रेंसिंग जैसे कि Skype, Facetime और Google Hangouts; यूजर द्वारा जनरेटेड वीडियो जैसे कि YouTube और मूवीज़ ऑन डिमांड जैसे कि Netflix; मल्टीप्लेयर ऑनलाइन गेम जैसे कि Second Life और World of Warcraft शामिल हैं। इसी अवधि के दौरान, हमने सोशल नेटवर्किंग एप्लीकेशन की एक नई पीढ़ी का उदय देखा है - जैसे कि Facebook, Instagram, Twitter और WeChat - जिसने इंटरनेट के नेटवर्क या राउटर और संचार लिंक के शीर्ष पर आकर्षक मानव नेटवर्क बनाए हैं। और हाल ही में, स्मार्टफोन के आगमन के साथ, स्थान आधारित मोबाइल ऐप्स की भरमार हो गई है, जिसमें लोकप्रिय चेक-इन, डेटिंग और सड़क यातायात पूर्वानुमान ऐप्स (जैसे येल्प, टिंडर, वाज़ और यिक याक) शामिल हैं। स्पष्ट रूप से, नए और रोमांचक इंटरनेट अनुप्रयोगों की गति में कोई कमी नहीं आई है। शायद इस पाठ के कुछ पाठक अगली पीढ़ी के किलर इंटरनेट एप्लिकेशन बनाएंगे!

इस अध्याय में हम नेटवर्क अनुप्रयोगों के वैचारिक और कार्यान्वयन पहलुओं का अध्ययन करेंगे। हम प्रमुख अनुप्रयोग-स्तर अवधारणाओं को परिभाषित करके शुरू करते हैं, जिसमें अनुप्रयोगों, क्लाइंट और सर्वर, प्रक्रियाओं और ट्रांसपोर्ट-लेयर इंटरफेस द्वारा आवश्यक नेटवर्क सेवाएँ शामिल हैं। हम वेब, ई-मेल, DNS, पीयर-टू-पीयर (P2P) फ़ाइल वितरण और वीडियो स्ट्रीमिंग सहित कई नेटवर्क अनुप्रयोगों की विस्तार से जाँच करते हैं।

(अध्याय 9 में स्ट्रीमिंग वीडियो और वीओआईपी सहित मल्टीमीडिया अनुप्रयोगों की आगे जांच की जाएगी।) फिर हम टीसीपी और यूडीपी दोनों पर नेटवर्क एप्लिकेशन डेवलपमेंट को कवर करते हैं। विशेष रूप से, हम सॉकेट इंटरफ़ेस का अध्ययन करते हैं और पायथन में कुछ सरल क्लाइंट-सर्वर अनुप्रयोगों के माध्यम से चलते हैं। हम अध्याय के अंत में कई मज़ेदार और दिलचस्प सॉकेट प्रोग्रामिंग असाइनमेंट भी प्रदान करते हैं।

प्रोटोकॉल के अध्ययन की शुरुआत करने के लिए एप्लीकेशन लेयर एक विशेष रूप से अच्छी जगह है। यह परिचित क्षेत्र है। हम उन कई अनुप्रयोगों से परिचित हैं जो उन प्रोटोकॉल पर निर्भर करते हैं जिनका हम अध्ययन करेंगे। इससे हमें प्रोटोकॉल के बारे में अच्छी जानकारी मिलेगी और हमें कई ऐसे मुद्दों से परिचित कराएगा जो हमें ट्रांसपोर्ट, नेटवर्क और लिंक लेयर प्रोटोकॉल का अध्ययन करते समय फिर से देखने को मिलेंगे।

2.1 नेटवर्क अनुप्रयोगों के सिद्धांत

मान लीजिए कि आपके पास एक नए नेटवर्क एप्लीकेशन का विचार है। शायद यह एप्लीकेशन मानवता के लिए एक महान सेवा होगी, या आपके प्रोफेसर को खुश करेगी, या आपको बहुत धन देगी, या बस इसे विकसित करना मज़ेदार होगा।
प्रेरणा चाहे जो भी हो, आइए अब देखें कि आप इस विचार को वास्तविक दुनिया के नेटवर्क अनुप्रयोग में कैसे परिवर्तित करते हैं।

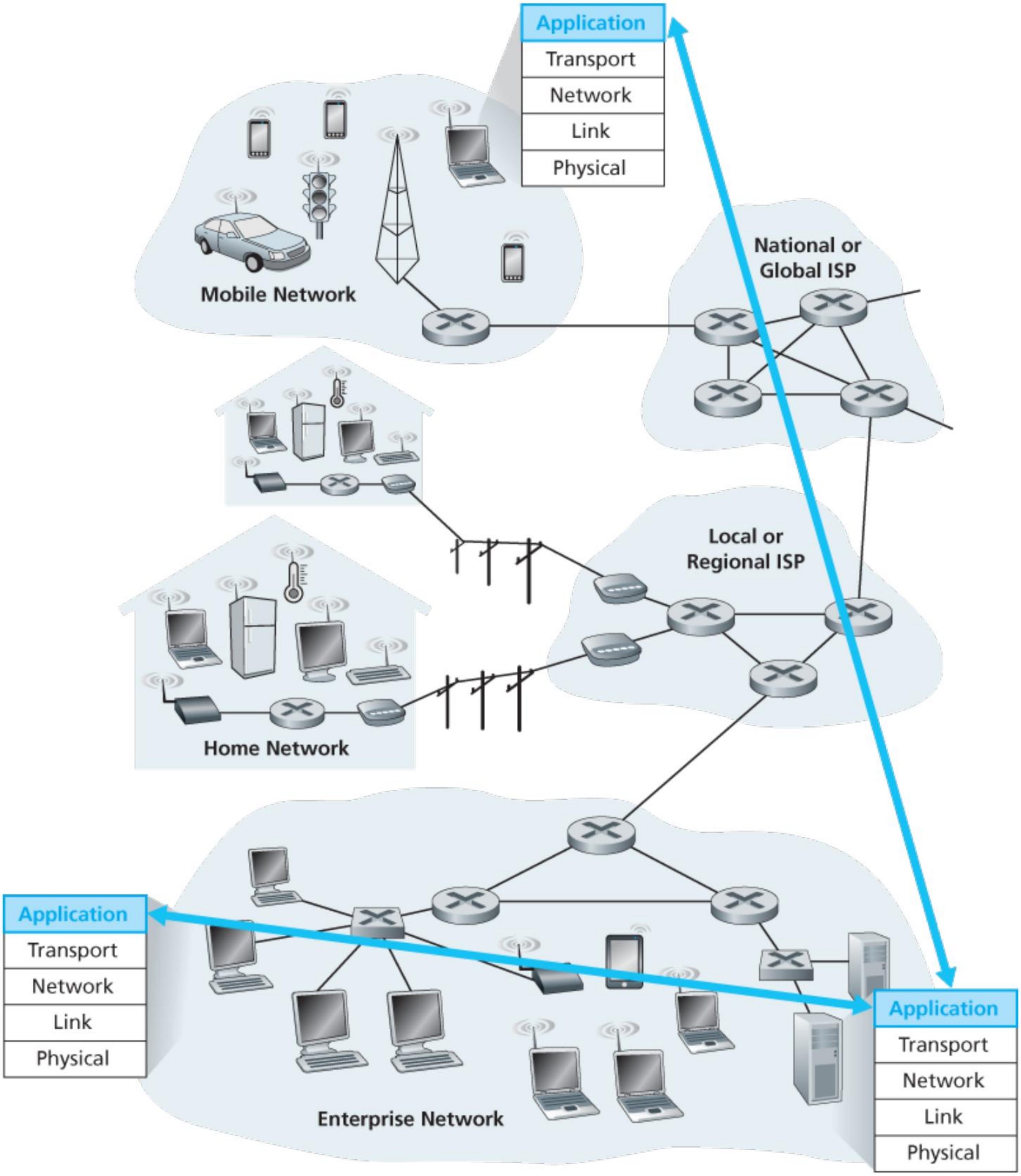
नेटवर्क एप्लिकेशन डेवलपमेंट का मूल उद्देश्य ऐसे प्रोग्राम लिखना है जो अलग-अलग एंड सिस्टम पर चलते हैं और नेटवर्क पर एक दूसरे से संवाद करते हैं। उदाहरण के लिए, वेब एप्लिकेशन में दो अलग-अलग प्रोग्राम होते हैं जो एक दूसरे से संवाद करते हैं: उपयोगकर्ता के होस्ट (डेस्कटॉप, लैपटॉप, टैबलेट, स्मार्टफोन, इत्यादि) में चलने वाला ब्राउज़र प्रोग्राम और वेब सर्वर होस्ट में चलने वाला वेब सर्वर प्रोग्राम। एक अन्य उदाहरण के रूप में, P2P फ़ाइल-शेयरिंग सिस्टम में प्रत्येक होस्ट में एक प्रोग्राम होता है जो फ़ाइल-शेयरिंग समुदाय में भाग लेता है। इस मामले में, विभिन्न होस्ट में प्रोग्राम समान हो सकते हैं

या समान.

इस प्रकार, जब आप अपना नया एप्लिकेशन विकसित कर रहे हों, तो आपको ऐसा सॉफ़्टवेयर लिखना होगा जो कई एंड सिस्टम पर चलेगा। यह सॉफ़्टवेयर, उदाहरण के लिए, C, Java या Python में लिखा जा सकता है। महत्वपूर्ण बात यह है कि आपको ऐसा सॉफ़्टवेयर लिखने की ज़रूरत नहीं है जो नेटवर्क-कोर डिवाइस, जैसे कि राउटर या लिंक-लेयर स्विच पर चलता हो। भले ही आप इन नेटवर्क-कोर डिवाइस के लिए एप्लिकेशन सॉफ़्टवेयर लिखना चाहें, लेकिन आप ऐसा नहीं कर पाएँगे।

जैसा कि हमने अध्याय 1 में सीखा था, और जैसा कि पहले चित्र 1.24 में दिखाया गया है, नेटवर्क-कोर डिवाइस अनुप्रयोग स्तर पर कार्य नहीं करते हैं, बल्कि इसके बजाय निचली परतों पर कार्य करते हैं - विशेष रूप से नेटवर्क स्तर पर और उससे नीचे।

इस मूल डिजाइन ने - अर्थात् अनुप्रयोग सॉफ्टवेयर को अंतिम प्रणालियों तक सीमित कर दिया है - जैसा कि चित्र 2.1 में दिखाया गया है, नेटवर्क अनुप्रयोगों की एक विशाल श्रृंखला के तीव्र विकास और परिनियोजन को सुगम बनाया है।



चित्र 2.1 नेटवर्क अनुप्रयोग के लिए संचार अनुप्रयोग स्तर पर अंतिम प्रणालियों के बीच होता है

2.1.1 नेटवर्क अनुप्रयोग आर्किटेक्चर

सॉफ़्टवेयर कोडिंग में उतरने से पहले, आपके पास अपने एप्लिकेशन के लिए एक व्यापक आर्किटेक्चरल प्लान होना चाहिए। ध्यान रखें कि एप्लिकेशन का आर्किटेक्चर नेटवर्क आर्किटेक्चर से बिल्कुल अलग होता है (उदाहरण के लिए,

पांच-परत इंटरनेट आर्किटेक्चर (अध्याय 1 में चर्चा की गई है)। एप्लिकेशन डेवलपर के दृष्टिकोण से, नेटवर्क आर्किटेक्चर तय है और अनुप्रयोगों को सेवाओं का एक विशिष्ट सेट प्रदान करता है ।

दूसरी ओर, आर्किटेक्चर **एप्लिकेशन डेवलपर द्वारा डिज़ाइन किया जाता है और यह निर्धारित करता है कि एप्लिकेशन कैसे काम करेगा।**

एप्लिकेशन को विभिन्न अंत प्रणालियों पर संरचित किया जाता है। एप्लिकेशन आर्किटेक्चर चुनने में,

एप्लिकेशन डेवलपर संभवतः आधुनिक नेटवर्क अनुप्रयोगों में प्रयुक्त दो प्रमुख वास्तुकला प्रतिमानों में से एक का उपयोग करेगा: क्लाइंट-सर्वर आर्किटेक्चर या पीयर-टू-पीयर (पी2पी) आर्किटेक्चर।

क्लाइंट-सर्वर आर्किटेक्चर में , हमेशा चालू रहने वाला एक होस्ट होता है, **जिसे** सर्वर कहा जाता है, जो कई अन्य होस्ट से अनुरोधों की सेवा करता है, जिन्हें क्लाइंट कहा जाता है। इसका एक क्लासिक उदाहरण वेब एप्लिकेशन है जिसके लिए हमेशा चालू रहने वाला वेब सर्वर क्लाइंट होस्ट पर चलने वाले ब्राउज़र से अनुरोधों की सेवा करता है। जब वेब सर्वर को क्लाइंट होस्ट से किसी ऑब्जेक्ट के लिए अनुरोध प्राप्त होता है, तो वह अनुरोधित ऑब्जेक्ट को क्लाइंट होस्ट को भेजकर प्रतिक्रिया करता है।

ध्यान दें कि क्लाइंट-सर्वर आर्किटेक्चर के साथ, क्लाइंट एक दूसरे के साथ सीधे संवाद नहीं करते हैं; उदाहरण के लिए, वेब एप्लिकेशन में, दो ब्राउज़र सीधे संवाद नहीं करते हैं। क्लाइंट-सर्वर आर्किटेक्चर की एक और विशेषता यह है कि सर्वर का एक निश्चित, प्रसिद्ध पता होता है, जिसे IP पता कहा जाता है (जिस पर हम जल्द ही चर्चा करेंगे)। क्योंकि सर्वर का एक निश्चित, प्रसिद्ध पता होता है, और क्योंकि सर्वर

हमेशा चालू रहने पर, क्लाइंट हमेशा सर्वर के आईपी पते पर पैकेट भेजकर सर्वर से संपर्क कर सकता है।

क्लाइंट-सर्वर आर्किटेक्चर वाले कुछ बेहतर ज्ञात अनुप्रयोगों में वेब, एफ़टीपी, टेलनेट शामिल हैं।

और ई-मेल। क्लाइंट-सर्वर आर्किटेक्चर चित्र 2.2(ए) में दिखाया गया है।

अक्सर क्लाइंट-सर्वर एप्लिकेशन में, एक सिंगल-सर्वर होस्ट क्लाइंट से सभी अनुरोधों को पूरा करने में असमर्थ होता है। उदाहरण के लिए, एक लोकप्रिय सोशल-नेटवर्किंग साइट जल्दी ही अभिभूत हो सकती है यदि उसके पास अपने सभी अनुरोधों को संभालने वाला केवल एक सर्वर हो। इस कारण से, एक डेटा सेंटर, जिसमें बड़ी संख्या में होस्ट होते हैं, का उपयोग अक्सर एक शक्तिशाली वर्चुअल सर्वर बनाने के लिए किया जाता है। सबसे लोकप्रिय इंटरनेट सेवाएँ - जैसे खोज इंजन (जैसे, Google, Bing, Baidu), इंटरनेट कॉमर्स (जैसे, Amazon, eBay, Alibaba), वेब-आधारित ई-मेल (जैसे, Gmail और Yahoo Mail), सोशल नेटवर्किंग (जैसे, Facebook, Instagram, Twitter, और

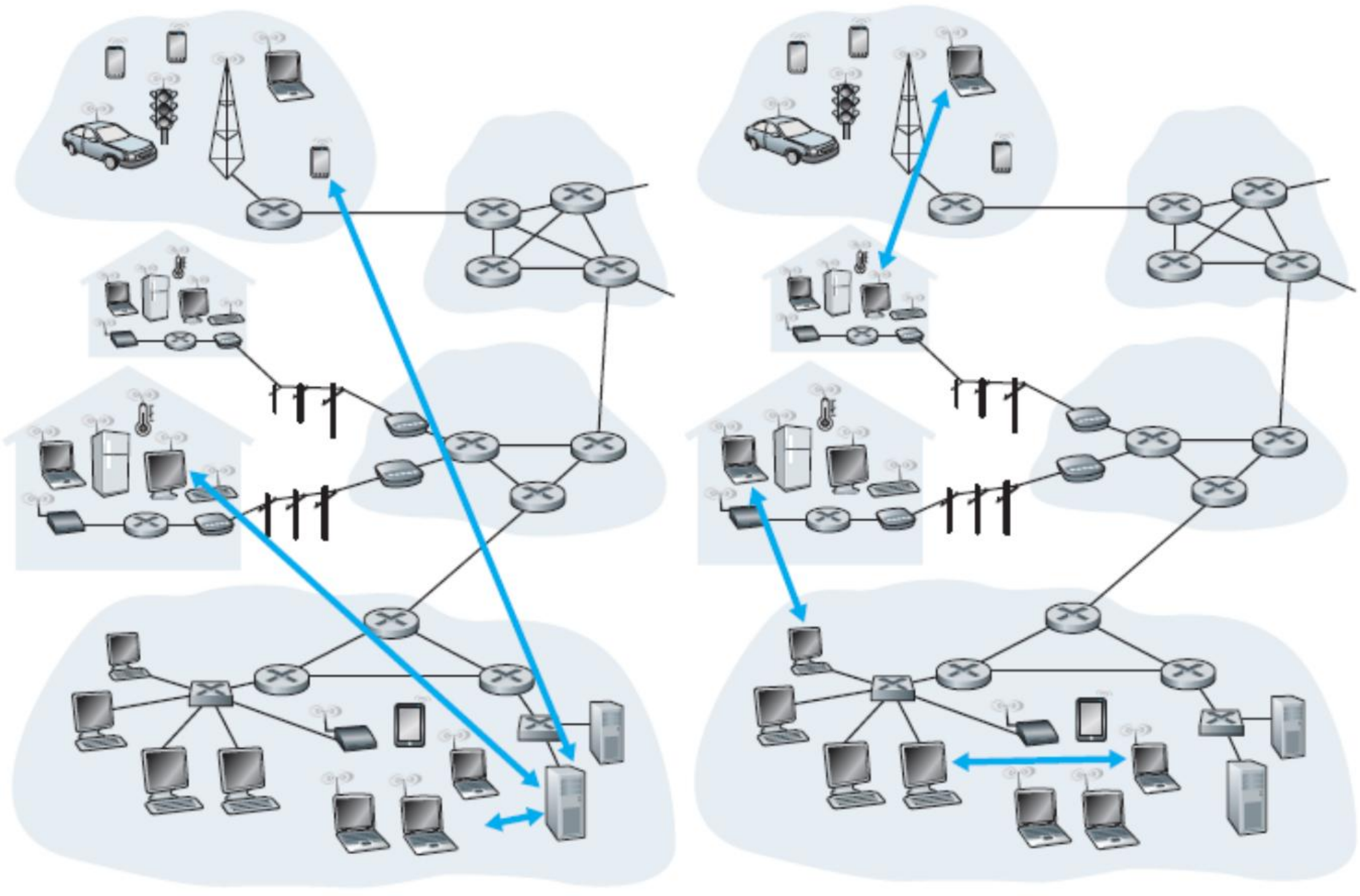
WeChat) - एक या अधिक डेटा सेंटर का उपयोग करते हैं। जैसा कि अनुभाग 1.3.3 में चर्चा की गई है, Google के दुनिया भर में 30 से 50 डेटा सेंटर हैं, जो सामूहिक रूप से खोज, YouTube, Gmail और अन्य सेवाओं को संभालते हैं। एक डेटा सेंटर में सैकड़ों हज़ारों सर्वर हो सकते हैं, जिन्हें पावर और पावर की आवश्यकता होती है।

इसके अतिरिक्त, सेवा प्रदाताओं को आवर्ती इंटरकनेक्शन और बैंडविड्थ लागत का भुगतान करना होगा

अपने डेटा केंद्रों से डेटा भेजने के लिए।

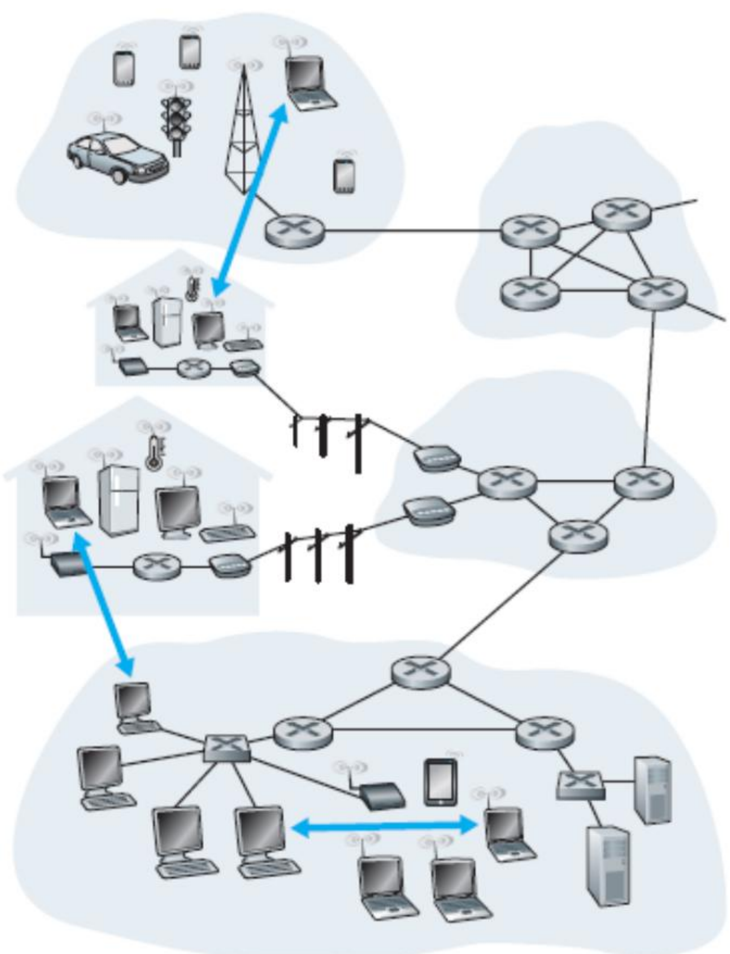
पी2पी आर्किटेक्चर में , डेटा सेंटर में समर्पित सर्वर पर न्यूनतम (या कोई) निर्भरता नहीं होती है। इसके बजाय, एप्लिकेशन रुक-रुक कर जुड़े होस्ट के जोड़े के बीच सीधे संचार का लाभ उठाता है, जिसे कहा जाता है

सहकर्मी। सहकर्मी सेवा प्रदाता के स्वामित्व में नहीं होते हैं, बल्कि वे डेस्कटॉप और लैपटॉप होते हैं जिन्हें उपयोगकर्ताओं द्वारा नियंत्रित किया जाता है, जिनमें से अधिकांश



a. Client-server architecture **b. Peer-to-peer architecture**

चित्र 2.2 (क) क्लाइंट-सर्वर आर्किटेक्चर; (ख) पी2पी आर्किटेक्चर



b. Peer-to-peer architecture

घरों, विश्वविद्यालयों और दफ़्तरों में रहने वाले सहकर्मी। चूँकि सहकर्मी किसी समर्पित सर्वर से गुज़रे बिना संवाद करते हैं, इसलिए इस आर्किटेक्चर को पीयर-टू-पीयर कहा जाता है। आज के सबसे लोकप्रिय और ट्रैफ़िक-गहन अनुप्रयोगों में से कई P2P आर्किटेक्चर पर आधारित हैं। इन अनुप्रयोगों में फ़ाइल शेयरिंग (जैसे, बिटटोरेंट), सहकर्मी-सहायता प्राप्त डाउनलोड त्वरण (जैसे, ज़ुनलेई), और इंटरनेट टेलीफोनी और वीडियो शामिल हैं

कॉन्फ्रेंस (जैसे, स्काइप) में P2P आर्किटेक्चर को दर्शाया गया है। हम उल्लेख करते हैं कि कुछ अनुप्रयोगों में हाइब्रिड आर्किटेक्चर होते हैं, जो क्लाइंट-सर्वर और P2P दोनों तत्वों को मिलाते हैं। उदाहरण के लिए, कई इंस्टेंट मैसेजिंग अनुप्रयोगों के लिए, सर्वर का उपयोग उपयोगकर्ताओं के आईपी पते को ट्रैक करने के लिए किया जाता है, लेकिन उपयोगकर्ता-से-उपयोगकर्ता संदेश सीधे उपयोगकर्ता होस्ट के बीच भेजे जाते हैं (मध्यवर्ती सर्वर से गुज़रे बिना)।

P2P आर्किटेक्चर की सबसे आकर्षक विशेषताओं में से एक उनकी स्व-मापनीयता है। उदाहरण के लिए, P2P फ़ाइल-शेयरिंग एप्लिकेशन में, हालांकि प्रत्येक सहकर्मी फ़ाइलों का अनुरोध करके कार्यभार उत्पन्न करता है, प्रत्येक सहकर्मी अन्य सहकर्मियों को फ़ाइलें वितरित करके सिस्टम में सेवा क्षमता भी जोड़ता है। P2P आर्किटेक्चर लागत प्रभावी भी हैं, क्योंकि उन्हें आम तौर पर महत्वपूर्ण सर्वर इन्फ्रास्ट्रक्चर और सर्वर बैंडविड्थ की आवश्यकता नहीं होती है (डेटासेंटर वाले क्लाइंट-सर्वर डिज़ाइन के विपरीत)। हालाँकि, P2P एप्लिकेशन को उनकी अत्यधिक विकेंद्रीकृत संरचना के कारण सुरक्षा, प्रदर्शन और विश्वसनीयता की चुनौतियों का सामना करना पड़ता है।

2.1.2 संचार प्रक्रियाएँ

अपना नेटवर्क एप्लीकेशन बनाने से पहले, आपको इस बात की बुनियादी समझ भी होनी चाहिए कि कई एंड सिस्टम में चलने वाले प्रोग्राम एक दूसरे से कैसे संवाद करते हैं। ऑपरेटिंग सिस्टम की शब्दावली में, यह वास्तव में प्रोग्राम नहीं बल्कि प्रक्रियाएँ **हैं** जो संवाद करती हैं। एक प्रक्रिया को एक प्रोग्राम के रूप में माना जा सकता है जो एक एंड सिस्टम के भीतर चल रहा है। जब प्रक्रियाएँ एक ही एंड सिस्टम पर चल रही होती हैं, तो वे एंड सिस्टम के ऑपरेटिंग सिस्टम द्वारा नियंत्रित नियमों का उपयोग करके इंटरप्रोसेस संचार के साथ एक दूसरे से संवाद कर सकती हैं। लेकिन इस पुस्तक में हम इस बात में विशेष रूप से रुचि नहीं रखते हैं कि प्रक्रियाएँ किस तरह से संवाद करती हैं।

एक ही होस्ट किस प्रकार संचार करता है, इसका अध्ययन नहीं किया गया है, बल्कि विभिन्न होस्टों (संभावित रूप से भिन्न ऑपरेटिंग सिस्टम के साथ) पर चलने वाली प्रक्रियाएं किस प्रकार **संचार करती हैं, इसका अध्ययन किया गया है।**

दो अलग-अलग एंड सिस्टम पर प्रक्रियाएँ कंप्यूटर नेटवर्क पर संदेशों **का आदान-प्रदान करके एक दूसरे के साथ संवाद करती हैं। एक भेजने वाली प्रक्रिया नेटवर्क में संदेश बनाती है और भेजती है; एक प्राप्त करने वाली प्रक्रिया नेटवर्क में संदेशों को भेजती है।**

प्रक्रिया इन संदेशों को प्राप्त करती है और संभवतः संदेशों को वापस भेजकर प्रतिक्रिया देती है। चित्र 2.1 दर्शाता है कि एक दूसरे के साथ संचार करने वाली प्रक्रियाएँ पाँच-परत प्रोटोकॉल स्टैक की एप्लिकेशन परत में रहती हैं।

क्लाइंट और सर्वर प्रक्रियाएँ

एक नेटवर्क अनुप्रयोग में प्रक्रियाओं के जोड़े होते हैं जो नेटवर्क पर एक दूसरे को संदेश भेजते हैं।

उदाहरण के लिए, वेब अनुप्रयोग में क्लाइंट ब्राउज़र प्रक्रिया वेब सर्वर के साथ संदेशों का आदान-प्रदान करती है

प्रक्रिया। P2P फ़ाइल-शेयरिंग सिस्टम में, एक फ़ाइल को एक पीयर में एक प्रक्रिया से दूसरे पीयर में एक प्रक्रिया में स्थानांतरित किया जाता है। संचार प्रक्रियाओं की प्रत्येक जोड़ी के लिए, हम आम तौर पर दो प्रक्रियाओं में से एक को लेबल करते हैं क्लाइंट और दूसरी प्रक्रिया सर्वर के रूप में। वेब के साथ, एक ब्राउज़र एक क्लाइंट प्रक्रिया है और एक वेब सर्वर एक सर्वर प्रक्रिया है। पी2पी फ़ाइल शेयरिंग के साथ, फ़ाइल डाउनलोड करने वाले सहकर्मी को क्लाइंट के रूप में लेबल किया जाता है, और फ़ाइल अपलोड करने वाले सहकर्मी को सर्वर के रूप में लेबल किया जाता है।

आपने देखा होगा कि कुछ अनुप्रयोगों में, जैसे कि P2P फ़ाइल शेयरिंग में, एक प्रक्रिया क्लाइंट और सर्वर दोनों हो सकती है। दरअसल, P2P फ़ाइल-शेयरिंग सिस्टम में एक प्रक्रिया फ़ाइलों को अपलोड और डाउनलोड दोनों कर सकती है।
फिर भी, प्रक्रियाओं की एक जोड़ी के बीच किसी भी दिए गए संचार सत्र के संदर्भ में, हम अभी भी एक प्रक्रिया को क्लाइंट और दूसरी प्रक्रिया को सर्वर के रूप में लेबल कर सकते हैं। हम क्लाइंट और सर्वर प्रक्रियाओं को इस प्रकार परिभाषित करते हैं:

> प्रक्रियाओं की एक जोड़ी के बीच संचार सत्र के संदर्भ में, संचार आरंभ करने वाली प्रक्रिया (अर्थात, सत्र की शुरुआत में दूसरी प्रक्रिया से संपर्क करती है) को क्लाइंट के रूप में लेबल किया जाता है। सत्र शुरू करने के लिए संपर्क किए जाने की प्रतीक्षा करने वाली प्रक्रिया सर्वर है।

वेब में, एक ब्राउज़र प्रक्रिया एक वेब सर्वर प्रक्रिया के साथ संपर्क आरंभ करती है; इसलिए ब्राउज़र प्रक्रिया क्लाइंट है और वेब सर्वर प्रक्रिया सर्वर है। पी2पी फ़ाइल शेयरिंग में, जब पीयर ए पीयर बी से एक विशिष्ट फ़ाइल भेजने के लिए कहता है, तो पीयर ए क्लाइंट होता है और पीयर बी इस विशिष्ट संचार सत्र के संदर्भ में सर्वर होता है। जब कोई भ्रम न हो, तो हम कभी-कभी "किसी एप्लिकेशन का क्लाइंट साइड और सर्वर साइड" शब्दावली का भी उपयोग करेंगे। इस अध्याय के अंत में, हम नेटवर्क एप्लिकेशन के क्लाइंट और सर्वर दोनों पक्षों के लिए सरल कोड के माध्यम से आगे बढ़ेंगे।

प्रक्रिया और कंप्यूटर नेटवर्क के बीच इंटरफ़ेस

जैसा कि ऊपर उल्लेख किया गया है, अधिकांश अनुप्रयोगों में संचार प्रक्रियाओं के जोड़े होते हैं, जिनमें से प्रत्येक जोड़ी में दो प्रक्रियाएँ एक-दूसरे को संदेश भेजती हैं। एक प्रक्रिया से दूसरी प्रक्रिया को भेजा गया कोई भी संदेश अंतर्निहित नेटवर्क से होकर गुजरना चाहिए। एक प्रक्रिया सॉकेट नामक सॉफ़्टवेयर इंटरफ़ेस के माध्यम से नेटवर्क में संदेश भेजती है और नेटवर्क से संदेश प्राप्त करती है। प्रक्रियाओं और सॉकेट को समझने में हमारी मदद करने के लिए आइए एक सादृश्य पर विचार करें। एक प्रक्रिया एक घर के समान है और इसका सॉकेट इसके दरवाज़े के समान है। जब एक प्रक्रिया किसी अन्य होस्ट पर किसी अन्य प्रक्रिया को संदेश भेजना चाहती है, तो वह संदेश को अपने दरवाज़े (सॉकेट) से बाहर धकेल देती है। यह भेजने वाली प्रक्रिया मानती है कि इसके दरवाज़े के दूसरी तरफ एक परिवहन अवसंरचना है जो संदेश को गंतव्य प्रक्रिया के दरवाज़े तक पहुँचाएगी। एक बार जब संदेश गंतव्य होस्ट पर पहुँच जाता है, तो संदेश प्राप्त करने वाली प्रक्रिया के दरवाज़े (सॉकेट) से होकर गुजरता है, और फिर प्राप्त करने वाली प्रक्रिया संदेश पर कार्य करती है।

चित्र 2.3 इंटरनेट पर संचार करने वाली दो प्रक्रियाओं के बीच सॉकेट संचार को दर्शाता है।
(चित्र 2.3 मानता है कि प्रक्रियाओं द्वारा उपयोग किया जाने वाला अंतर्निहित परिवहन प्रोटोकॉल इंटरनेट का TCP प्रोटोकॉल है।) जैसा कि इस चित्र में दिखाया गया है, सॉकेट होस्ट के भीतर एप्लिकेशन लेयर और ट्रांसपोर्ट लेयर के बीच का इंटरफ़ेस है। इसे एप्लिकेशन प्रोग्रामिंग इंटरफ़ेस (API) भी कहा जाता है।

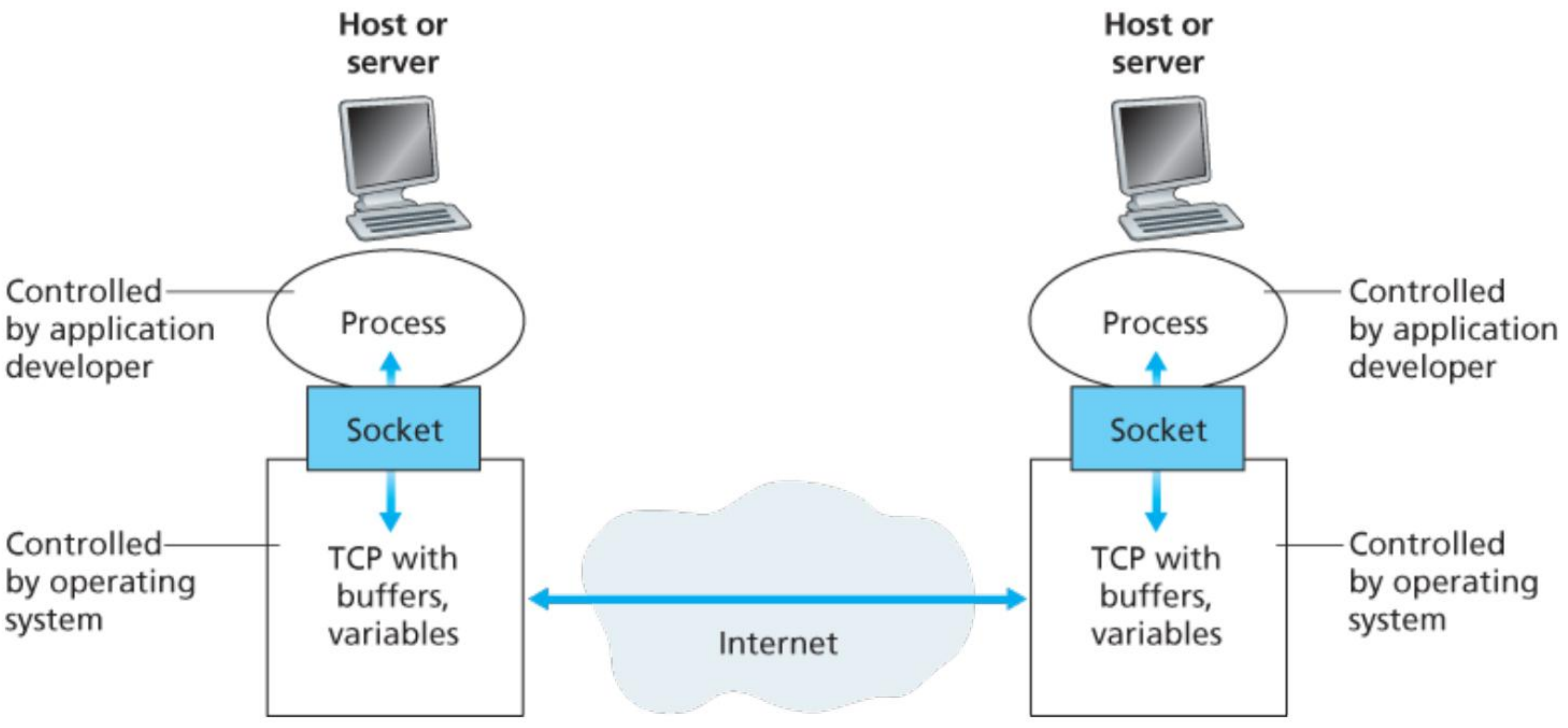
एप्लिकेशन और नेटवर्क के बीच, चूंकि सॉकेट प्रोग्रामिंग इंटरफ़ेस है जिसके साथ नेटवर्क एप्लिकेशन बनाए जाते हैं। एप्लिकेशन डेवलपर के पास सॉकेट के एप्लिकेशन-लेयर साइड पर सब कुछ का नियंत्रण होता है, लेकिन सॉकेट के ट्रांसपोर्ट-लेयर साइड पर उसका बहुत कम नियंत्रण होता है। ट्रांसपोर्ट-लेयर साइड पर एप्लिकेशन डेवलपर के पास एकमात्र नियंत्रण (1) ट्रांसपोर्ट प्रोटोकॉल का विकल्प और (2) शायद कुछ ट्रांसपोर्ट-लेयर पैरामीटर जैसे कि अधिकतम बफर और अधिकतम को ठीक करने की क्षमता है।

सेगमेंट आकार (अध्याय 3 में कवर किया जाएगा)। एक बार जब एप्लिकेशन डेवलपर ट्रांसपोर्ट प्रोटोकॉल चुनता है (यदि कोई विकल्प उपलब्ध है), तो एप्लिकेशन को द्वारा प्रदान की गई ट्रांसपोर्ट-लेयर सेवाओं का उपयोग करके बनाया जाता है

उस प्रोटोकॉल का उपयोग करें। हम अनुभाग 2.7 में सॉकेट्स के बारे में विस्तार से जानेंगे ।

संबोधित प्रक्रियाएं

किसी विशेष गंतव्य स्थान पर डाक भेजने के लिए, गंतव्य स्थान का पता होना आवश्यक है।
इसी प्रकार, एक होस्ट पर चल रही प्रक्रिया द्वारा किसी अन्य होस्ट पर चल रही प्रक्रिया को पैकेट भेजने के लिए, प्राप्तकर्ता प्रक्रिया के पास पता होना आवश्यक है।



चित्र 2.3 अनुप्रयोग प्रक्रियाएँ, सॉकेट और अंतर्निहित परिवहन प्रोटोकॉल

प्राप्त करने की प्रक्रिया की पहचान करने के लिए, दो जानकारी को निर्दिष्ट करने की आवश्यकता है: (1) होस्ट का पता और (2) एक पहचानकर्ता जो गंतव्य होस्ट में प्राप्त करने की प्रक्रिया को निर्दिष्ट करता है।

इंटरनेट पर होस्ट की पहचान उसके आईपी पते से होती है। हम आईपी पते के बारे में विस्तार से चर्चा करेंगे।

अध्याय 4. अभी के लिए, हमें बस इतना जानना है कि IP पता एक 32-बिट मात्रा है जिसे हम होस्ट की विशिष्ट पहचान के रूप में सोच सकते हैं। संदेश जिस होस्ट को भेजा जाना है, उसका पता जानने के अलावा, भेजने की प्रक्रिया को होस्ट में चल रही प्राप्त करने वाली प्रक्रिया (अधिक विशेष रूप से, प्राप्त करने वाली सॉकेट) की भी पहचान करनी चाहिए। यह जानकारी इसलिए ज़रूरी है क्योंकि आम तौर पर एक होस्ट कई नेटवर्क एप्लिकेशन चला सकता है। एक गंतव्य पोर्ट नंबर इस उद्देश्य को पूरा करता है। लोकप्रिय एप्लिकेशन

विशिष्ट पोर्ट नंबर असाइन किए गए हैं। उदाहरण के लिए, एक वेब सर्वर की पहचान पोर्ट नंबर 80 से होती है। एक मेल सर्वर प्रक्रिया (SMTP प्रोटोकॉल का उपयोग करके) की पहचान पोर्ट नंबर 25 से होती है। जाने-माने पोर्ट की सूची

सभी इंटरनेट मानक प्रोटोकॉल के नंबर www.iana.org पर देखे जा सकते हैं। हम अध्याय 3 में पोर्ट नंबरों की विस्तार से जांच करेंगे ।

2.1.3 अनुप्रयोगों के लिए उपलब्ध परिवहन सेवाएँ

स्मरण रखें कि सॉकेट अनुप्रयोग प्रक्रिया और ट्रांसपोर्ट-लेयर प्रोटोकॉल के बीच का इंटरफ़ेस है।

भेजने वाले पक्ष पर मौजूद एप्लिकेशन सॉकेट के माध्यम से संदेश भेजता है। सॉकेट के दूसरी तरफ, ट्रांसपोर्ट लेयर प्रोटोकॉल की जिम्मेदारी होती है कि वह संदेशों को सॉकेट तक पहुंचाए।

प्राप्त करने की प्रक्रिया.

इंटरनेट समेत कई नेटवर्क एक से ज़्यादा ट्रांसपोर्ट लेयर प्रोटोकॉल उपलब्ध कराते हैं।

किसी एप्लिकेशन को विकसित करने के लिए, आपको उपलब्ध ट्रांसपोर्ट-लेयर प्रोटोकॉल में से एक चुनना होगा। आप यह चुनाव कैसे करेंगे? सबसे अधिक संभावना है कि आप उपलब्ध ट्रांसपोर्ट-लेयर द्वारा प्रदान की जाने वाली सेवाओं का अध्ययन करेंगे

प्रोटोकॉल चुनें, और फिर उन सेवाओं के साथ प्रोटोकॉल चुनें जो आपके एप्लिकेशन की ज़रूरतों से सबसे अच्छी तरह मेल खाते हों।

यह स्थिति दो शहरों के बीच यात्रा के लिए ट्रेन या हवाई जहाज़ परिवहन में से किसी एक को चुनने के समान है। आपको इनमें से किसी एक को चुनना होगा, और प्रत्येक परिवहन मोड अलग-अलग सेवाएँ प्रदान करता है। (उदाहरण के लिए, ट्रेन शहर के बीच से पिक-अप और ड्रॉप-ऑफ़ की सुविधा देती है, जबकि विमान कम समय में यात्रा करने की सुविधा देता है।)

ट्रांसपोर्ट लेयर प्रोटोकॉल इसे लागू करने वाले अनुप्रयोगों को कौन सी सेवाएँ प्रदान कर सकता है? हम संभावित सेवाओं को मोटे तौर पर चार आयामों के आधार पर वर्गीकृत कर सकते हैं: विश्वसनीय डेटा स्थानांतरण, थ्रूपुट, टाइमिंग,

और सुरक्षा.

विश्वसनीय डेटा स्थानांतरण

जैसा कि अध्याय 1 में चर्चा की गई है , पैकेट कंप्यूटर नेटवर्क के भीतर खो सकते हैं। उदाहरण के लिए, एक पैकेट राउटर में बफर को ओवरफ्लो कर सकता है, या इसके कुछ बिट्स दूषित होने के बाद होस्ट या राउटर द्वारा त्याग दिया जा सकता है। कई अनुप्रयोगों के लिए - जैसे इलेक्ट्रॉनिक मेल, फ़ाइल स्थानांतरण, रिमोट होस्ट एक्सेस, वेब दस्तावेज़ स्थानांतरण और वित्तीय अनुप्रयोग - डेटा हानि के विनाशकारी परिणाम हो सकते हैं (बाद के मामले में, बैंक या ग्राहक के लिए!)। इस प्रकार, इन अनुप्रयोगों का समर्थन करने के लिए, यह गारंटी देने के लिए कुछ किया जाना चाहिए कि अनुप्रयोग के एक छोर से भेजा गया डेटा अनुप्रयोग के दूसरे छोर पर सही ढंग से और पूरी तरह से वितरित किया जाए। यदि कोई प्रोटोकॉल ऐसी गारंटीकृत डेटा डिलीवरी सेवा प्रदान करता है, तो इसे विश्वसनीय डेटा ट्रांसफर प्रदान करने वाला कहा जाता है। एक महत्वपूर्ण सेवा जो ट्रांसपोर्ट-लेयर प्रोटोकॉल संभावित रूप से किसी अनुप्रयोग को प्रदान कर सकता है, वह है प्रक्रिया-दर-प्रक्रिया विश्वसनीय डेटा ट्रांसफर। जब एक ट्रांसपोर्ट प्रोटोकॉल यह सेवा प्रदान करता है, तो भेजने वाली प्रक्रिया बस अपना डेटा सॉकेट में पास कर सकती है और पूरे विश्वास के साथ जान सकती है कि डेटा प्राप्त करने वाली प्रक्रिया में त्रुटियों के बिना पहुंचेगा।

जब ट्रांसपोर्ट लेयर प्रोटोकॉल विश्वसनीय डेटा स्थानांतरण प्रदान नहीं करता है, तो प्रोटोकॉल द्वारा भेजा गया कुछ डेटा विश्वसनीय डेटा स्थानांतरण प्रदान नहीं करता है।

हो सकता है कि भेजने की प्रक्रिया कभी भी प्राप्त करने की प्रक्रिया तक न पहुँचे। यह हानि-सहिष्णु अनुप्रयोगों के लिए स्वीकार्य हो सकता है, विशेष रूप से मल्टीमीडिया अनुप्रयोग जैसे कि संवादात्मक ऑडियो/वीडियो जो कुछ मात्रा में डेटा हानि को सहन कर सकते हैं। इन मल्टीमीडिया अनुप्रयोगों में, खोए हुए डेटा के परिणामस्वरूप ऑडियो/वीडियो में एक छोटी सी गड़बड़ी हो सकती है - कोई महत्वपूर्ण हानि नहीं।

प्रवाह

अध्याय 1 में हमने उपलब्ध थ्रूपुट की अवधारणा पेश की, जो नेटवर्क पथ पर दो प्रक्रियाओं के बीच संचार सत्र के संदर्भ में, वह दर है जिस पर भेजने वाली प्रक्रिया प्राप्त करने वाली प्रक्रिया को बिट्स वितरित कर सकती है। क्योंकि अन्य सत्र नेटवर्क पथ पर बैंडविड्थ साझा करेंगे, और क्योंकि ये अन्य सत्र आते-जाते रहेंगे, उपलब्ध थ्रूपुट समय के साथ उतार-चढ़ाव कर सकता है। ये अवलोकन एक अन्य प्राकृतिक सेवा की ओर ले जाते हैं जो एक ट्रांसपोर्ट-लेयर प्रोटोकॉल प्रदान कर सकता है, अर्थात्, कुछ निर्दिष्ट दर पर गारंटीकृत उपलब्ध थ्रूपुट। इस तरह के साथ

सेवा के लिए, एप्लिकेशन r बिट्स/सेकंड की गारंटीकृत थ्रूपुट का अनुरोध कर सकता है, और फिर ट्रांसपोर्ट प्रोटोकॉल यह सुनिश्चित करेगा कि उपलब्ध थ्रूपुट हमेशा कम से कम r बिट्स/सेकंड हो। ऐसी गारंटीकृत थ्रूपुट सेवा कई एप्लिकेशन को आकर्षित करेगी। उदाहरण के लिए, यदि कोई इंटरनेट टेलीफोनी एप्लिकेशन 32 kbps पर आवाज़ को एनकोड करता है, तो उसे नेटवर्क में डेटा भेजने और इस दर पर प्राप्त करने वाले एप्लिकेशन को डेटा डिलीवर करने की आवश्यकता होती है। यदि ट्रांसपोर्ट प्रोटोकॉल यह थ्रूपुट प्रदान नहीं कर सकता है, तो एप्लिकेशन को कम दर पर एनकोड करने की आवश्यकता होगी (और इस कम कोडिंग दर को बनाए रखने के लिए पर्याप्त थ्रूपुट प्राप्त करना होगा) या इसे छोड़ना पड़ सकता है, क्योंकि आवश्यक थ्रूपुट का आधा प्राप्त करना इस इंटरनेट टेलीफोनी एप्लिकेशन के लिए बहुत कम या कोई उपयोग नहीं है। ऐसे एप्लिकेशन जिनमें थ्रूपुट की आवश्यकता होती है, उन्हें बैंडविड्थ-संवेदनशील एप्लिकेशन कहा जाता है। कई मौजूदा मल्टीमीडिया एप्लिकेशन बैंडविड्थ संवेदनशील हैं, हालांकि कुछ मल्टीमीडिया एप्लिकेशन वर्तमान में उपलब्ध थ्रूपुट से मेल खाने वाली दर पर डिजिटाइज्ड आवाज़ या वीडियो को एनकोड करने के लिए अनुकूली कोडिंग तकनीकों का उपयोग कर सकते हैं।

जबकि बैंडविड्थ-संवेदनशील अनुप्रयोगों में विशिष्ट थ्रूपुट आवश्यकताएँ होती हैं, लोचदार अनुप्रयोग जितना उपलब्ध हो उतना अधिक या कम थ्रूपुट का उपयोग कर सकते हैं। इलेक्ट्रॉनिक मेल, फ़ाइल ट्रांसफ़र और वेब ट्रांसफ़र सभी लोचदार अनुप्रयोग हैं। बेशक, जितना ज़्यादा थ्रूपुट होगा, उतना ही बेहतर होगा। एक कहावत है जो कहती है कि कोई व्यक्ति बहुत अमीर, बहुत पतला या बहुत ज़्यादा थ्रूपुट वाला नहीं हो सकता!

समय

ट्रांसपोर्ट-लेयर प्रोटोकॉल टाइमिंग गारंटी भी प्रदान कर सकता है। थ्रूपुट गारंटी की तरह, टाइमिंग गारंटी कई आकार और रूपों में आ सकती है। एक उदाहरण गारंटी यह हो सकती है कि प्रेषक द्वारा सॉकेट में पंप किया गया प्रत्येक बिट रिसीवर के सॉकेट पर 100 एमएसईसी से अधिक समय बाद नहीं पहुंचेगा। ऐसी सेवा इंटरेक्टिव रीयल-टाइम अनुप्रयोगों, जैसे कि इंटरनेट टेलीफोनी, वर्चुअल वातावरण, टेलीकॉन्फ्रेंसिंग और मल्टीप्लेयर गेम के लिए आकर्षक होगी, जिनमें से सभी को सख्त समय की बाधाओं की आवश्यकता होती है

डेटा डिलीवरी को प्रभावी बनाने के लिए। (अध्याय 9 देखें, [गौथियर 1999; रामजी 1994]।) उदाहरण के लिए, इंटरनेट टेलीफोनी में लंबे विलंब के कारण बातचीत में अस्वाभाविक रुकावटें आती हैं; मल्टीप्लेयर गेम या वर्चुअल इंटरैक्टिव वातावरण में, कार्रवाई करने और प्रतिक्रिया देखने के बीच लंबा विलंब होता है

पर्यावरण से (उदाहरण के लिए, एंड-टू-एंड कनेक्शन के अंत में किसी अन्य प्लेयर से) एप्लिकेशन को कम यथार्थवादी लगता है। गैर-वास्तविक-समय अनुप्रयोगों के लिए, कम देरी हमेशा बेहतर होती है

उच्चतर विलंब, लेकिन अंत-से-अंत विलंब पर कोई सख्त प्रतिबंध नहीं लगाया गया है।

सुरक्षा

अंत में, ट्रांसपोर्ट प्रोटोकॉल किसी एप्लिकेशन को एक या अधिक सुरक्षा सेवाएँ प्रदान कर सकता है। उदाहरण के लिए, भेजने वाले होस्ट में, ट्रांसपोर्ट प्रोटोकॉल भेजने की प्रक्रिया द्वारा प्रेषित सभी डेटा को एन्क्रिप्ट कर सकता है, और प्राप्त करने वाले होस्ट में, ट्रांसपोर्ट-लेयर प्रोटोकॉल डेटा को प्राप्त करने वाली प्रक्रिया को वितरित करने से पहले डेटा को डिक्रिप्ट कर सकता है। ऐसी सेवा दो प्रक्रियाओं के बीच गोपनीयता प्रदान करेगी, भले ही डेटा को भेजने और प्राप्त करने वाली प्रक्रियाओं के बीच किसी तरह से देखा गया हो। एक ट्रांसपोर्ट प्रोटोकॉल गोपनीयता के अलावा अन्य सुरक्षा सेवाएँ भी प्रदान कर सकता है, जिसमें डेटा अखंडता और अंत-बिंदु शामिल हैं

प्रमाणीकरण, ऐसे विषय जिन पर हम अध्याय 8 में विस्तार से चर्चा करेंगे ।

2.1.4 इंटरनेट द्वारा प्रदान की जाने वाली परिवहन सेवाएँ

इस बिंदु तक, हम उन परिवहन सेवाओं पर विचार कर रहे थे जो एक कंप्यूटर नेटवर्क **सामान्य रूप से प्रदान** कर सकता है । आइए अब अधिक विशिष्ट बनें और इंटरनेट द्वारा प्रदान की जाने वाली परिवहन सेवाओं के प्रकार की जांच करें। इंटरनेट (और, अधिक सामान्यतः, TCP/IP नेटवर्क) अनुप्रयोगों के लिए दो परिवहन प्रोटोकॉल उपलब्ध कराता है, UDP और TCP। जब आप (एक एप्लिकेशन डेवलपर के रूप में) इंटरनेट के लिए एक नया नेटवर्क एप्लिकेशन बनाते हैं, तो सबसे पहले आपको यह निर्णय लेना होता है कि UDP या TCP का उपयोग करना है या नहीं। इनमें से प्रत्येक

ये प्रोटोकॉल इनवोकिंग एप्लीकेशन को सेवाओं का एक अलग सेट प्रदान करते हैं। चित्र 2.4 कुछ चयनित एप्लीकेशन के लिए सेवा आवश्यकताओं को दर्शाता है।

टीसीपी सेवाएँ

टीसीपी सेवा मॉडल में एक कनेक्शन-उन्मुख सेवा और एक विश्वसनीय डेटा स्थानांतरण सेवा शामिल है।

जब कोई एप्लिकेशन TCP को अपने ट्रांसपोर्ट प्रोटोकॉल के रूप में आमंत्रित करता है, तो एप्लिकेशन को ये दोनों प्राप्त होते हैं टीसीपी से सेवाएँ.

- कनेक्शन-उन्मुख सेवा। TCP में क्लाइंट और सर्वर एक दूसरे के साथ ट्रांसपोर्ट-लेयर नियंत्रण जानकारी का आदान-प्रदान करते हैं, इससे पहले कि एप्लिकेशन-स्तरीय संदेश प्रवाहित होने लगें। यह तथाकथित हैंडशेकिंग प्रक्रिया क्लाइंट और सर्वर को सचेत करती है, जिससे वे पैकेटों के हमले के लिए तैयार हो जाते हैं। हैंडशेकिंग चरण के बाद, सॉकेट्स के बीच **एक** TCP कनेक्शन **मौजूद माना जाता है**

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps—1Mbps Video: 10 kbps—5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps—10 kbps	Yes: 100s of msec
Smartphone messaging	No loss	Elastic	Yes and no

चित्र 2.4 चयनित नेटवर्क अनुप्रयोगों की आवश्यकताएँ

दो प्रक्रियाओं का कनेक्शन। यह कनेक्शन एक पूर्ण-द्वैध कनेक्शन है जिसमें दो प्रक्रियाएँ एक ही समय में कनेक्शन पर एक दूसरे को संदेश भेज सकती हैं। जब एप्लिकेशन समाप्त हो जाता है

संदेश भेजने के लिए, इसे कनेक्शन को तोड़ना होगा। अध्याय 3 में हम कनेक्शन-उन्मुख सेवा पर विस्तार से चर्चा करेंगे और जांच करेंगे कि इसे कैसे लागू किया जाता है।

- विश्वसनीय डेटा ट्रांसफर सेवा। संचार प्रक्रियाएँ बिना किसी त्रुटि के और उचित क्रम में भेजे गए सभी डेटा को वितरित करने के लिए TCP पर भरोसा कर सकती हैं। जब एप्लिकेशन का एक पक्ष सॉकेट में बाइट्स की एक धारा को पास करता है, तो वह प्राप्त करने वाले सॉकेट को बाइट्स की समान धारा वितरित करने के लिए TCP पर भरोसा कर सकता है,

कोई भी बाइट गायब या डुप्लिकेट नहीं होनी चाहिए।

टीसीपी में एक भीड़-भाड़ नियंत्रण तंत्र भी शामिल है, जो संचार प्रक्रियाओं के प्रत्यक्ष लाभ के बजाय इंटरनेट के सामान्य कल्याण के लिए एक सेवा है। टीसीपी भीड़-भाड़ नियंत्रण तंत्र एक भेजने वाली प्रक्रिया (क्लाइंट या सर्वर) को तब धीमा कर देता है जब नेटवर्क के बीच भीड़भाड़ होती है

प्रेषक और प्राप्तकर्ता। जैसा कि हम देखेंगे

सुरक्षा पर ध्यान दें

टीसीपी को सुरक्षित करना

न तो TCP और न ही UDP कोई एन्क्रिप्शन प्रदान करता है - जो डेटा भेजने की प्रक्रिया अपने सॉकेट में भेजती है, वही डेटा नेटवर्क पर गंतव्य प्रक्रिया तक जाता है। इसलिए, उदाहरण के लिए, यदि भेजने की प्रक्रिया अपने सॉकेट में क्लियरटेक्स्ट (यानी, अनएन्क्रिप्टेड) में पासवर्ड भेजती है, तो क्लियरटेक्स्ट पासवर्ड प्रेषक और रिसीवर के बीच सभी लिंक पर जाएगा, संभावित रूप से किसी भी बीच के लिंक पर सूँघकर खोजा जा सकता है। चूँकि गोपनीयता और अन्य सुरक्षा मुद्दे कई अनुप्रयोगों के लिए महत्वपूर्ण हो गए हैं, इसलिए इंटरनेट समुदाय ने TCP के लिए एक संवर्द्धन विकसित किया है, जिसे सिक्योर सॉकेट लेयर (SSL) कहा जाता है। SSL के साथ TCP-संवर्धित न केवल

> वह सब कुछ करता है जो पारंपरिक TCP करता है, लेकिन महत्वपूर्ण प्रक्रिया-दर-प्रक्रिया सुरक्षा भी प्रदान करता है
>
> एन्क्रिप्शन, डेटा अखंडता और अंतिम बिंदु प्रमाणीकरण सहित सेवाएँ। हम इस बात पर ज़ोर देते हैं कि
>
> SSL, TCP और UDP के समान स्तर पर तीसरा इंटरनेट ट्रांसपोर्ट प्रोटोकॉल नहीं है, बल्कि यह TCP का संवर्द्धन है, जिसमें संवर्द्धन को एप्लीकेशन लेयर में लागू किया जा रहा है। विशेष रूप से, यदि कोई एप्लीकेशन SSL की सेवाओं का उपयोग करना चाहता है, तो उसे एप्लीकेशन के क्लाइंट और सर्वर दोनों तरफ SSL कोड (मौजूदा, अत्यधिक अनुकूलित लाइब्रेरी और क्लास) शामिल करने की आवश्यकता होती है। SSL का अपना सॉकेट API है जो पारंपरिक TCP सॉकेट API के समान है। जब कोई एप्लीकेशन SSL का उपयोग करता है, तो भेजने की प्रक्रिया SSL सॉकेट में क्लियरटेक्स्ट डेटा पास करती है; भेजने वाले होस्ट में SSL फिर डेटा को एन्क्रिप्ट करता है और एन्क्रिप्ट किए गए डेटा को TCP सॉकेट में पास करता है।
>
> एन्क्रिप्ट किया गया डेटा इंटरनेट पर रिसीविंग प्रक्रिया में TCP सॉकेट तक जाता है। रिसीविंग सॉकेट एन्क्रिप्ट किए गए डेटा को SSL में भेजता है, जो डेटा को डिक्रिप्ट करता है। अंत में, SSL अपने SSL सॉकेट के माध्यम से क्लियरटेक्स्ट डेटा को रिसीविंग प्रक्रिया में भेजता है। हम SSL को विस्तार से कवर करेंगे
>
> अध्याय 8 में कुछ विस्तार से बताया गया है ।

अध्याय 3 में , TCP संकुलन नियंत्रण भी प्रत्येक TCP कनेक्शन को नेटवर्क बैंडविड्थ के उचित हिस्से तक सीमित करने का प्रयास करता है।

यूडीपी सेवाएं

यूडीपी एक बिना किसी तामझाम वाला, हल्का ट्रांसपोर्ट प्रोटोकॉल है, जो न्यूनतम सेवाएँ प्रदान करता है। यूडीपी कनेक्शन रहित है, इसलिए दो प्रक्रियाओं के बीच संचार शुरू होने से पहले कोई हैंडशेकिंग नहीं होती है। यूडीपी एक अविश्वसनीय प्रदान करता है

डेटा ट्रांसफ़र सेवा - यानी, जब कोई प्रक्रिया UDP सॉकेट में संदेश भेजती है, तो UDP इस बात की **कोई गारंटी** नहीं **देता** कि संदेश कभी भी प्राप्त करने वाली प्रक्रिया तक पहुँचेगा। इसके अलावा, जो संदेश प्राप्त करने वाली प्रक्रिया तक पहुँचते हैं, वे क्रम से नहीं पहुँच सकते।

यूडीपी में भीड़-भाड़ नियंत्रण तंत्र शामिल नहीं है, इसलिए यूडीपी का भेजने वाला पक्ष अपनी इच्छानुसार किसी भी दर पर नीचे की परत (नेटवर्क परत) में डेटा पंप कर सकता है। (हालांकि, ध्यान दें कि वास्तविक अंत-से-अंत थ्रूपुट बीच की लिंक की सीमित संचरण क्षमता या भीड़भाड़ के कारण इस दर से कम हो सकता है)।

इंटरनेट ट्रांसपोर्ट प्रोटोकॉल द्वारा प्रदान न की जाने वाली सेवाएँ

हमने ट्रांसपोर्ट प्रोटोकॉल सेवाओं को चार आयामों के साथ व्यवस्थित किया है: विश्वसनीय डेटा ट्रांसफ़र, थ्रूपुट, टाइमिंग और सुरक्षा। इनमें से कौन सी सेवाएँ TCP और UDP द्वारा प्रदान की जाती हैं? हमने पहले ही नोट कर लिया है कि TCP विश्वसनीय एंड-टू-एंड डेटा ट्रांसफ़र प्रदान करता है। और हम यह भी जानते हैं कि सुरक्षा सेवाएँ प्रदान करने के लिए SSL के साथ एप्लिकेशन लेयर पर TCP को आसानी से बढ़ाया जा सकता है। लेकिन TCP और UDP के हमारे संक्षिप्त विवरण में

यूडीपी में, थ्रूपुट या टाइमिंग गारंटी का कोई उल्लेख नहीं था - आज के इंटरनेट ट्रांसपोर्ट प्रोटोकॉल द्वारा प्रदान की जाने वाली सेवाएँ नहीं । क्या इसका मतलब यह है कि समय-संवेदनशील अनुप्रयोग जैसे कि

क्या आज के इंटरनेट में इंटरनेट टेलीफोनी नहीं चल सकती? इसका उत्तर स्पष्ट रूप से नहीं है - इंटरनेट कई वर्षों से समय-संवेदनशील अनुप्रयोगों की मेजबानी कर रहा है। ये अनुप्रयोग अक्सर काफी अच्छी तरह से काम करते हैं क्योंकि

उन्हें इस गारंटी की कमी से निपटने के लिए यथासंभव अधिकतम सीमा तक डिज़ाइन किया गया है।

अध्याय 9 में इनमें से कई डिज़ाइन तरकीबों की जाँच करें। फिर भी, जब देरी अत्यधिक होती है, या एंड-टू-एंड थ्रूपुट सीमित होता है, तो चतुर डिज़ाइन की अपनी सीमाएँ होती हैं। संक्षेप में, आज का इंटरनेट अक्सर समय-संवेदनशील अनुप्रयोगों को संतोषजनक सेवा प्रदान कर सकता है, लेकिन यह कोई समय या थ्रूपुट प्रदान नहीं कर सकता है

गारंटी.

चित्र 2.5 कुछ लोकप्रिय इंटरनेट अनुप्रयोगों द्वारा उपयोग किए जाने वाले परिवहन प्रोटोकॉल को दर्शाता है। हम देखते हैं कि ई-मेल, रिमोट टर्मिनल एक्सेस, वेब और फ़ाइल ट्रांसफ़र सभी TCP का उपयोग करते हैं। इन अनुप्रयोगों ने मुख्य रूप से TCP को चुना है क्योंकि TCP विश्वसनीय डेटा ट्रांसफ़र प्रदान करता है, यह गारंटी देता है कि सभी डेटा अंततः अपने गंतव्य तक पहुँच जाएगा। चूँकि इंटरनेट टेलीफ़ोनी अनुप्रयोग (जैसे स्काइप) अक्सर कुछ नुकसान सहन कर सकते हैं लेकिन प्रभावी होने के लिए न्यूनतम दर की आवश्यकता होती है, इंटरनेट टेलीफ़ोनी अनुप्रयोगों के डेवलपर्स आमतौर पर अपने अनुप्रयोगों को UDP पर चलाना पसंद करते हैं, जिससे TCP की भीड़ नियंत्रण प्रणाली और पैकेट ओवरहेड्स को दरकिनार किया जा सके। लेकिन चूँकि कई फ़ायरवॉल UDP ट्रैफ़िक (अधिकांश प्रकार के) को ब्लॉक करने के लिए कॉन्फ़िगर किए गए हैं, इसलिए इंटरनेट टेलीफ़ोनी अनुप्रयोगों को अक्सर UDP संचार विफल होने पर बैकअप के रूप में TCP का उपयोग करने के लिए डिज़ाइन किया जाता है।

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

चित्र 2.5 लोकप्रिय इंटरनेट अनुप्रयोग, उनके अनुप्रयोग-स्तर प्रोटोकॉल और उनके अंतर्निहित परिवहन प्रोटोकॉल

2.1.5 अनुप्रयोग-स्तर प्रोटोकॉल

हमने अभी सीखा है कि नेटवर्क प्रक्रियाएं संदेश भेजकर एक दूसरे से संवाद करती हैं।

सॉकेट। लेकिन ये संदेश कैसे संरचित हैं? संदेशों में विभिन्न क्षेत्रों के क्या अर्थ हैं? प्रक्रियाएँ संदेश कब भेजती हैं? ये प्रश्न हमें एप्लिकेशन-लेयर प्रोटोकॉल के दायरे में लाते हैं। एक एप्लिकेशन-लेयर प्रोटोकॉल परिभाषित करता है कि कैसे एक एप्लिकेशन की प्रक्रियाएँ, विभिन्न एंड सिस्टम पर चल रही हैं, एक दूसरे को संदेश भेजती हैं। विशेष रूप से, एक एप्लिकेशन-लेयर प्रोटोकॉल परिभाषित करता है:

- आदान-प्रदान किए जाने वाले संदेशों के प्रकार, उदाहरण के लिए, अनुरोध संदेश और प्रतिक्रिया संदेश
- विभिन्न संदेश प्रकारों का वाक्यविन्यास, जैसे संदेश में फ़ील्ड और फ़ील्ड को किस प्रकार चित्रित किया जाता है

- क्षेत्रों का शब्दार्थ, अर्थात क्षेत्रों में मौजूद जानकारी का अर्थ
- यह निर्धारित करने के नियम कि कोई प्रक्रिया कब और कैसे संदेश भेजती है तथा संदेशों पर प्रतिक्रिया देती है

कुछ एप्लिकेशन-लेयर प्रोटोकॉल RFC में निर्दिष्ट हैं और इसलिए वे सार्वजनिक डोमेन में हैं।

उदाहरण के लिए, वेब का एप्लिकेशन-लेयर प्रोटोकॉल, HTTP (हाइपरटेक्स्ट ट्रांसफ़र प्रोटोकॉल [RFC 2616]), RFC के रूप में उपलब्ध है। यदि कोई ब्राउज़र डेवलपर HTTP RFC के नियमों का पालन करता है, तो ब्राउज़र किसी भी वेब सर्वर से वेब पेज प्राप्त करने में सक्षम होगा जिसने HTTP RFC के नियमों का भी पालन किया है। कई

अन्य एप्लिकेशन-लेयर प्रोटोकॉल मालिकाना हैं और जानबूझकर सार्वजनिक डोमेन में उपलब्ध नहीं हैं।

उदाहरण के लिए, स्काइप मालिकाना अनुप्रयोग-स्तर प्रोटोकॉल का उपयोग करता है।

नेटवर्क अनुप्रयोगों और अनुप्रयोग-स्तर प्रोटोकॉल के बीच अंतर करना महत्वपूर्ण है।

एप्लीकेशन-लेयर प्रोटोकॉल नेटवर्क एप्लीकेशन का केवल एक हिस्सा है (हालांकि, हमारे दृष्टिकोण से एप्लीकेशन का एक बहुत ही महत्वपूर्ण हिस्सा!)। आइए कुछ उदाहरण देखें। वेब एक क्लाइंट-सर्वर एप्लीकेशन है जो उपयोगकर्ताओं को मांग पर वेब सर्वर से दस्तावेज़ प्राप्त करने की अनुमति देता है। वेब एप्लीकेशन में कई घटक होते हैं, जिसमें दस्तावेज़ प्रारूपों (यानी, HTML), वेब ब्राउज़र (उदाहरण के लिए, फ़ायरफ़ॉक्स और माइक्रोसॉफ़्ट इंटरनेट एक्सप्लोरर), वेब सर्वर (उदाहरण के लिए, अपाचे और माइक्रोसॉफ़्ट सर्वर) और एक एप्लीकेशन-लेयर प्रोटोकॉल के लिए एक मानक शामिल है। वेब का एप्लीकेशन-लेयर प्रोटोकॉल, HTTP, ब्राउज़र और वेब सर्वर के बीच आदान-प्रदान किए जाने वाले संदेशों के प्रारूप और अनुक्रम को परिभाषित करता है। इस प्रकार, HTTP वेब एप्लीकेशन का केवल एक हिस्सा है (हालांकि, एक महत्वपूर्ण हिस्सा)। एक अन्य उदाहरण के रूप में, एक

इंटरनेट ई-मेल एप्लीकेशन में भी कई घटक होते हैं, जिनमें मेल सर्वर शामिल होते हैं जो उपयोगकर्ता को ईमेल भेजते हैं।

मेलबॉक्स; मेल क्लाइंट (जैसे कि माइक्रोसॉफ्ट आउटलुक) जो उपयोगकर्ताओं को संदेश पढ़ने और बनाने की अनुमति देते हैं; ई-मेल संदेश की संरचना को परिभाषित करने के लिए एक मानक; और अनुप्रयोग-स्तर प्रोटोकॉल जो परिभाषित करते हैं कि कैसे

सर्वरों के बीच संदेश कैसे पारित किए जाते हैं, सर्वरों और मेल क्लाइंटों के बीच संदेश कैसे पारित किए जाते हैं,

और संदेश हेडर की सामग्री की व्याख्या कैसे की जानी है। मुख्य अनुप्रयोग-स्तर प्रोटोकॉल

इलेक्ट्रॉनिक मेल के लिए सबसे महत्वपूर्ण प्रोटोकॉल SMTP (सिंपल मेल ट्रांसफर प्रोटोकॉल) [RFC 5321] है। इस प्रकार, ई-मेल का मुख्य एप्लिकेशन-लेयर प्रोटोकॉल, SMTP, ई-मेल एप्लिकेशन का केवल एक हिस्सा (यद्यपि एक महत्वपूर्ण हिस्सा) है।

2.1.6 इस पुस्तक में शामिल नेटवर्क अनुप्रयोग

हर दिन नए सार्वजनिक डोमेन और मालिकाना इंटरनेट एप्लिकेशन विकसित किए जा रहे हैं।

बड़ी संख्या में इंटरनेट अनुप्रयोगों को विश्वकोशीय तरीके से कवर करते हुए, हमने कुछ ऐसे अनुप्रयोगों पर ध्यान केंद्रित करना चुना है जो व्यापक और महत्वपूर्ण दोनों हैं। इस अध्याय में हम पाँच पर चर्चा करते हैं

महत्वपूर्ण अनुप्रयोग: वेब, इलेक्ट्रॉनिक मेल, निर्देशिका सेवा वीडियो स्ट्रीमिंग, और पी 2 पी

अनुप्रयोग। हम सबसे पहले वेब पर चर्चा करते हैं, न केवल इसलिए कि यह एक बहुत लोकप्रिय अनुप्रयोग है, बल्कि इसलिए भी कि इसका अनुप्रयोग-स्तर प्रोटोकॉल, HTTP, सीधा और समझने में आसान है। फिर हम इलेक्ट्रॉनिक मेल पर चर्चा करते हैं, जो इंटरनेट का पहला किलर अनुप्रयोग है। ई-मेल वेब की तुलना में अधिक जटिल है।

यह समझ में आता है कि यह एक नहीं बल्कि कई एप्लिकेशन-लेयर प्रोटोकॉल का उपयोग करता है। ईमेल के बाद, हम DNS को कवर करते हैं, जो इंटरनेट के लिए एक निर्देशिका सेवा प्रदान करता है। अधिकांश उपयोगकर्ता सीधे DNS के साथ बातचीत नहीं करते हैं; इसके बजाय, उपयोगकर्ता अन्य अनुप्रयोगों (वेब, फ़ाइल स्थानांतरण और इलेक्ट्रॉनिक मेल सहित) के माध्यम से अप्रत्यक्ष रूप से DNS का आह्वान करते हैं। DNS अच्छी तरह से दिखाता है कि कोर नेटवर्क कार्यक्षमता का एक हिस्सा (नेटवर्क-नाम से नेटवर्क-पता अनुवाद) इंटरनेट में एप्लिकेशन लेयर पर कैसे लागू किया जा सकता है। फिर हम P2P फ़ाइल शेयरिंग अनुप्रयोगों पर चर्चा करते हैं, और मांग पर वीडियो स्ट्रीमिंग पर चर्चा करके अपने एप्लिकेशन अध्ययन को पूरा करते हैं,

इसमें कंटेंट डिस्ट्रीब्यूशन नेटवर्क पर संग्रहित वीडियो वितरित करना शामिल है। अध्याय 9 में, हम मल्टीमीडिया अनुप्रयोगों को अधिक गहराई से कवर करेंगे, जिसमें वॉयस ओवर आईपी और वीडियो कॉन्फ्रेंसिंग शामिल है।

2.2 वेब और HTTP

1990 के दशक की शुरुआत तक इंटरनेट का इस्तेमाल मुख्य रूप से शोधकर्ताओं, शिक्षाविदों और विश्वविद्यालय के छात्रों द्वारा रिमोट होस्ट में लॉग इन करने, स्थानीय होस्ट से रिमोट होस्ट में फ़ाइलें स्थानांतरित करने और इसके विपरीत, समाचार प्राप्त करने और भेजने और इलेक्ट्रॉनिक मेल प्राप्त करने और भेजने के लिए किया जाता था। हालाँकि ये अनुप्रयोग अत्यंत उपयोगी थे (और अभी भी हैं), इंटरनेट अनिवार्य रूप से अकादमिक और शोध समुदायों के बाहर अज्ञात था। फिर, 1990 के दशक की शुरुआत में, एक प्रमुख नया अनुप्रयोग सामने आया- वर्ल्ड वाइड

वेब [बर्नर्स-ली 1994]। वेब पहला इंटरनेट एप्लीकेशन था जिसने आम जनता का ध्यान खींचा। इसने नाटकीय रूप से बदल दिया, और बदलना जारी है, कि लोग अपने कार्य वातावरण के अंदर और बाहर कैसे बातचीत करते हैं। इसने इंटरनेट को कई डेटा नेटवर्क में से सिर्फ़ एक से बढ़ाकर अनिवार्य रूप से एकमात्र डेटा नेटवर्क बना दिया।

शायद जो बात उपयोगकर्ताओं को सबसे ज़्यादा आकर्षित करती है, वह यह है कि वेब मांग पर काम करता है। उपयोगकर्ता जो चाहते हैं, उसे वे तब प्राप्त करते हैं जब वे चाहते हैं। यह पारंपरिक प्रसारण रेडियो और टेलीविज़न से अलग है, जो उपयोगकर्ताओं को तब सुनने के लिए बाध्य करते हैं जब सामग्री प्रदाता सामग्री उपलब्ध कराता है। मांग पर उपलब्ध होने के अलावा, वेब में कई अन्य अद्भुत विशेषताएँ हैं जिन्हें लोग पसंद करते हैं और संजोते हैं। किसी भी व्यक्ति के लिए वेब पर जानकारी उपलब्ध कराना बेहद आसान है - हर कोई बेहद कम लागत पर प्रकाशक बन सकता है। हाइपरलिंक और सर्च इंजन हमें जानकारी के महासागर में नेविगेट करने में मदद करते हैं।

फोटो और वीडियो हमारी इंद्रियों को उत्तेजित करते हैं। फॉर्म, जावास्क्रिप्ट, जावा एप्लेट और कई अन्य डिवाइस हमें पेजों और साइटों के साथ बातचीत करने में सक्षम बनाते हैं। और वेब और इसके प्रोटोकॉल YouTube, वेब-आधारित ई-मेल (जैसे जीमेल) और इंस्टाग्राम और गूगल मैप्स सहित अधिकांश मोबाइल इंटरनेट एप्लिकेशन के लिए एक प्लेटफ़ॉर्म के रूप में काम करते हैं।

2.2.1 HTTP का अवलोकन

हाइपरटेक्स्ट ट्रांसफर प्रोटोकॉल (HTTP), वेब का एप्लिकेशन-लेयर प्रोटोकॉल, इसके मूल में है।

वेब। इसे [RFC 1945] और [RFC 2616] में परिभाषित किया गया है। HTTP को दो प्रोग्राम में लागू किया जाता है: एक क्लाइंट प्रोग्राम और एक सर्वर प्रोग्राम। क्लाइंट प्रोग्राम और सर्वर प्रोग्राम, अलग-अलग एंड सिस्टम पर निष्पादित होते हैं, HTTP संदेशों का आदान-प्रदान करके एक-दूसरे से बात करते हैं। HTTP इन संदेशों की संरचना को परिभाषित करता है और क्लाइंट और सर्वर संदेशों का आदान-प्रदान कैसे करते हैं। HTTP को विस्तार से समझाने से पहले, हमें कुछ वेब शब्दावली की समीक्षा करनी चाहिए।

वेब पेज (जिसे दस्तावेज़ भी कहा जाता है) में ऑब्जेक्ट होते हैं। ऑब्जेक्ट बस एक फ़ाइल होती है - जैसे कि HTML फ़ाइल, JPEG इमेज, जावा एप्लेट या वीडियो क्लिप - जिसे एक ही URL द्वारा संबोधित किया जा सकता है। अधिकांश वेब पेज में एक बेस HTML फ़ाइल और कई संदर्भित ऑब्जेक्ट होते हैं। उदाहरण के लिए, यदि कोई वेब पेज

यदि HTML टेक्स्ट और पाँच JPEG इमेज हैं, तो वेब पेज में छह ऑब्जेक्ट हैं: बेस HTML फ़ाइल और पाँच इमेज। बेस HTML फ़ाइल पेज में मौजूद अन्य ऑब्जेक्ट को ऑब्जेक्ट के URL के साथ संदर्भित करती है।

प्रत्येक URL के दो घटक होते हैं: उस सर्वर का होस्टनाम जिसमें ऑब्जेक्ट स्थित होता है और ऑब्जेक्ट का पथ नाम. उदाहरण के लिए, यूआरएल

```
http://www.someSchool.edu/someDepartment/picture.gif
```

होस्टनाम के लिए www.someSchool.edu **और** पथ नाम के लिए /someDepartment/picture.gif है । **क्योंकि** वेब ब्राउज़र (जैसे कि इंटरनेट एक्सप्लोरर और फ़ायरफ़ॉक्स) क्लाइंट साइड को लागू करते हैं

HTTP, वेब के संदर्भ में, हम ब्राउज़र और क्लाइंट **शब्दों का** परस्पर उपयोग करेंगे। वेब सर्वर, जो HTTP के सर्वर साइड को लागू करते हैं, वेब ऑब्जेक्ट्स को रखते हैं, जिनमें से प्रत्येक को URL द्वारा संबोधित किया जा सकता है।

लोकप्रिय वेब सर्वरों में अपाचे और माइक्रोसॉफ्ट इंटरनेट इंफॉर्मेशन सर्वर शामिल हैं।

HTTP यह परिभाषित करता है कि वेब क्लाइंट किस प्रकार वेब सर्वर से वेब पेजों का अनुरोध करते हैं और सर्वर किस प्रकार वेब सर्वर को वेब पेजों को स्थानांतरित करते हैं।

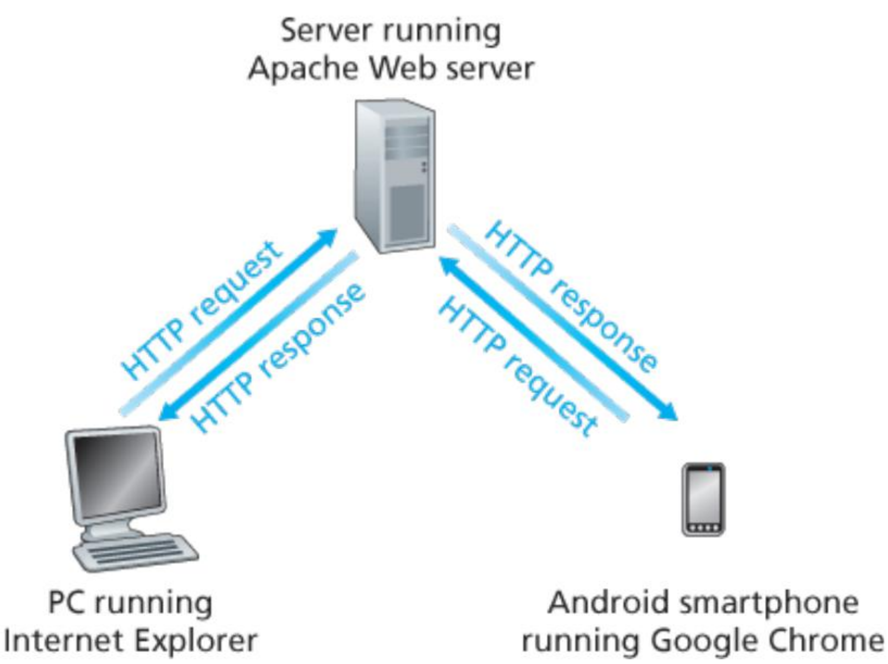
पेज को क्लाइंट में ट्रांसफर करना। हम क्लाइंट और सर्वर के बीच की बातचीत पर बाद में विस्तार से चर्चा करेंगे, लेकिन सामान्य तौर पर

इस विचार को चित्र 2.6 में दर्शाया गया है । जब कोई उपयोगकर्ता किसी वेब पेज का अनुरोध करता है (उदाहरण के लिए, हाइपरलिंक पर क्लिक करता है), तो ब्राउज़र सर्वर को पेज में मौजूद ऑब्जेक्ट के लिए HTTP अनुरोध संदेश भेजता है।

अनुरोध प्राप्त करता है और HTTP प्रतिक्रिया संदेशों के साथ प्रतिक्रिया करता है जिसमें ऑब्जेक्ट्स शामिल होते हैं।

HTTP अपने अंतर्निहित ट्रांसपोर्ट प्रोटोकॉल के रूप में TCP का उपयोग करता है (UDP के शीर्ष पर चलने के बजाय)। HTTP क्लाइंट सबसे पहले सर्वर के साथ TCP कनेक्शन आरंभ करता है। एक बार कनेक्शन स्थापित हो जाने के बाद, ब्राउज़र

और सर्वर प्रक्रियाएँ अपने सॉकेट इंटरफ़ेस के माध्यम से TCP तक पहुँचती हैं। जैसा कि अनुभाग 2.1 में वर्णित है, क्लाइंट साइड पर सॉकेट इंटरफ़ेस क्लाइंट प्रक्रिया और TCP कनेक्शन के बीच का द्वार है; सर्वर साइड पर यह सर्वर प्रक्रिया और TCP कनेक्शन के बीच का द्वार है। क्लाइंट अपने सॉकेट इंटरफ़ेस में HTTP अनुरोध संदेश भेजता है और अपने सॉकेट इंटरफ़ेस से HTTP प्रतिक्रिया संदेश प्राप्त करता है। इसी तरह, HTTP सर्वर अनुरोध संदेश प्राप्त करता है



चित्र 2.6 HTTP अनुरोध-प्रतिक्रिया व्यवहार

अपने सॉकेट इंटरफ़ेस से संदेश भेजता है और अपने सॉकेट इंटरफ़ेस में प्रतिक्रिया संदेश भेजता है। एक बार जब क्लाइंट अपने सॉकेट इंटरफ़ेस में संदेश भेजता है, तो संदेश क्लाइंट के हाथों से बाहर निकल जाता है और TCP के "हाथों" में चला जाता है।

सेक्शन 2.1 से याद करें कि TCP HTTP को एक विश्वसनीय डेटा ट्रांसफर सेवा प्रदान करता है। इसका मतलब है कि क्लाइंट प्रक्रिया द्वारा भेजा गया प्रत्येक HTTP अनुरोध संदेश अंततः सर्वर पर बरकरार पहुंचता है; इसी तरह,
सर्वर प्रक्रिया द्वारा भेजा गया प्रत्येक HTTP प्रतिक्रिया संदेश अंततः क्लाइंट तक बरकरार पहुंचता है। यहाँ
हम लेयर्ड आर्किटेक्चर के एक बड़े लाभ को देखते हैं - HTTP को खोए हुए डेटा या TCP द्वारा नेटवर्क के भीतर डेटा के नुकसान या पुनर्व्यवस्था से उबरने के विवरण के बारे में चिंता करने की ज़रूरत नहीं है। यह TCP और प्रोटोकॉल स्टैक की निचली परतों में प्रोटोकॉल का काम है।

यह ध्यान रखना महत्वपूर्ण है कि सर्वर क्लाइंट के बारे में कोई भी स्टेट जानकारी संग्रहीत किए बिना क्लाइंट को अनुरोधित फ़ाइलें भेजता है। यदि कोई
विशेष क्लाइंट कुछ समय की अवधि में एक ही ऑब्जेक्ट को दो बार मांगता है
कुछ सेकंड के बाद, सर्वर यह कहकर प्रतिक्रिया नहीं देता कि उसने अभी-अभी क्लाइंट को ऑब्जेक्ट दिया है; इसके बजाय,
सर्वर ऑब्जेक्ट को फिर से भेजता है, क्योंकि वह पूरी तरह से भूल गया है कि उसने पहले क्या किया था। क्योंकि HTTP सर्वर क्लाइंट के बारे में कोई जानकारी नहीं रखता है, इसलिए HTTP को स्टेटलेस प्रोटोकॉल कहा जाता है। हम यह भी टिप्पणी करते हैं

वेब क्लाइंट-सर्वर एप्लीकेशन आर्किटेक्चर का उपयोग करता है, जैसा कि अनुभाग 2.1 में वर्णित है। एक वेब सर्वर हमेशा चालू रहता है, एक निश्चित आईपी पते के साथ, और यह संभावित रूप से लाखों अलग-अलग अनुरोधों की सेवा करता है
ब्राउज़र.

2.2.2 गैर-स्थायी और स्थायी कनेक्शन

कई इंटरनेट अनुप्रयोगों में, क्लाइंट और सर्वर एक विस्तारित अवधि के लिए संचार करते हैं, जिसमें क्लाइंट अनुरोधों की एक श्रृंखला बनाता है और सर्वर प्रत्येक अनुरोध का जवाब देता है। एप्लिकेशन और एप्लिकेशन के उपयोग के तरीके के आधार पर, अनुरोधों की श्रृंखला बैक-टू-बैक बनाई जा सकती है,

नियमित अंतराल पर या रुक-रुक कर। जब यह क्लाइंट-सर्वर इंटरैक्शन हो रहा हो
टीसीपी, एप्लिकेशन डेवलपर को एक महत्वपूर्ण निर्णय लेने की आवश्यकता है - क्या प्रत्येक अनुरोध/प्रतिक्रिया
क्या सभी अनुरोधों और उनके संगत प्रतिक्रियाओं को एक ही TCP कनेक्शन पर भेजा जाना चाहिए ? पहले दृष्टिकोण में, एप्लिकेशन को गैर-स्थायी कनेक्शन का उपयोग करने के लिए कहा जाता है; और दूसरे दृष्टिकोण में, स्थायी कनेक्शन। एक गहरी समझ हासिल करने के लिए

इस डिज़ाइन समस्या को समझने के लिए, आइए एक विशिष्ट एप्लिकेशन, अर्थात् HTTP, के संदर्भ में स्थायी कनेक्शन के लाभ और हानि की जांच करें, जो गैर-स्थायी कनेक्शन और स्थायी कनेक्शन दोनों का उपयोग कर सकता है। हालाँकि HTTP अपने डिफ़ॉल्ट मोड में स्थायी कनेक्शन का उपयोग करता है, लेकिन HTTP क्लाइंट और सर्वर को इसके बजाय गैर-स्थायी कनेक्शन का उपयोग करने के लिए कॉन्फ़िगर किया जा सकता है।

गैर-स्थायी कनेक्शन के साथ HTTP

आइए गैर-स्थायी कनेक्शन के मामले में सर्वर से क्लाइंट तक वेब पेज को स्थानांतरित करने के चरणों के माध्यम से चलें। मान लें कि पेज में एक बेस HTML फ़ाइल और 10 JPEG इमेज हैं, और ये सभी 11 ऑब्जेक्ट एक ही सर्वर पर रहते हैं। इसके अलावा मान लें कि बेस HTML फ़ाइल के लिए URL

है

```
http://www.someSchool.edu/someDepartment/home.index
```

यहां बताया गया है कि क्या होता है:

1. HTTP क्लाइंट प्रक्रिया सर्वर www.someSchool.edu पर **एक TCP कनेक्शन आरंभ करती है** पोर्ट नंबर 80, जो HTTP के लिए डिफ़ॉल्ट पोर्ट नंबर है। TCP कनेक्शन से जुड़ा हुआ, क्लाइंट पर एक सॉकेट और सर्वर पर एक सॉकेट होगा।
2. HTTP क्लाइंट अपने सॉकेट के माध्यम से सर्वर को HTTP अनुरोध संदेश भेजता है। संदेश में पथ नाम /someDepartment/home .index **शामिल है** । (हम नीचे HTTP संदेशों पर विस्तार से चर्चा करेंगे।)
3. HTTP सर्वर प्रक्रिया अपने सॉकेट के माध्यम से अनुरोध संदेश प्राप्त करती है, ऑब्जेक्ट को पुनः प्राप्त करती है /someDepartment/home.index अपने स्टोरेज (RAM या डिस्क) से, ऑब्जेक्ट को इसमें समाहित करता है एक HTTP प्रतिक्रिया संदेश, और प्रतिक्रिया संदेश को अपने सॉकेट के माध्यम से क्लाइंट को भेजता है।
4. HTTP सर्वर प्रक्रिया TCP को TCP कनेक्शन बंद करने के लिए कहती है। (लेकिन TCP तब तक कनेक्शन समाप्त नहीं करता जब तक कि उसे यह पता न चल जाए कि क्लाइंट को प्रतिक्रिया संदेश सही सलामत प्राप्त हो गया है।)
5. HTTP क्लाइंट को प्रतिक्रिया संदेश प्राप्त होता है। TCP कनेक्शन समाप्त हो जाता है। संदेश यह संकेत देता है कि एनकैप्सुलेटेड ऑब्जेक्ट एक HTML फ़ाइल है। क्लाइंट प्रतिक्रिया संदेश से फ़ाइल निकालता है, HTML फ़ाइल की जाँच करता है, और 10 JPEG ऑब्जेक्ट के संदर्भ पाता है।
6. फिर प्रत्येक संदर्भित JPEG ऑब्जेक्ट के लिए पहले चार चरण दोहराए जाते हैं।

जैसे ही ब्राउज़र वेब पेज प्राप्त करता है, वह उपयोगकर्ता को पेज दिखाता है। दो अलग-अलग ब्राउज़र एक वेब पेज को कुछ अलग तरीकों से समझ सकते हैं (यानी, उपयोगकर्ता को दिखा सकते हैं)। HTTP का इससे कोई लेना-देना नहीं है

क्लाइंट द्वारा वेब पेज की व्याख्या कैसे की जाती है। HTTP विनिर्देश ([RFC 1945] और [RFC 2616]) केवल क्लाइंट HTTP प्रोग्राम और सर्वर HTTP के बीच संचार प्रोटोकॉल को परिभाषित करते हैं

कार्यक्रम.

ऊपर दिए गए चरण गैर-स्थायी कनेक्शन के उपयोग को दर्शाते हैं, जहाँ सर्वर द्वारा ऑब्जेक्ट भेजे जाने के बाद प्रत्येक TCP कनेक्शन बंद हो जाता है - अन्य ऑब्जेक्ट के लिए कनेक्शन कायम नहीं रहता है। ध्यान दें कि प्रत्येक TCP कनेक्शन बंद हो जाता है।

कनेक्शन केवल एक अनुरोध संदेश और एक प्रतिक्रिया संदेश स्थानांतरित करता है। इस प्रकार, इस उदाहरण में,

जब कोई उपयोगकर्ता वेब पेज का अनुरोध करता है, तो 11 टीसीपी कनेक्शन उत्पन्न होते हैं।

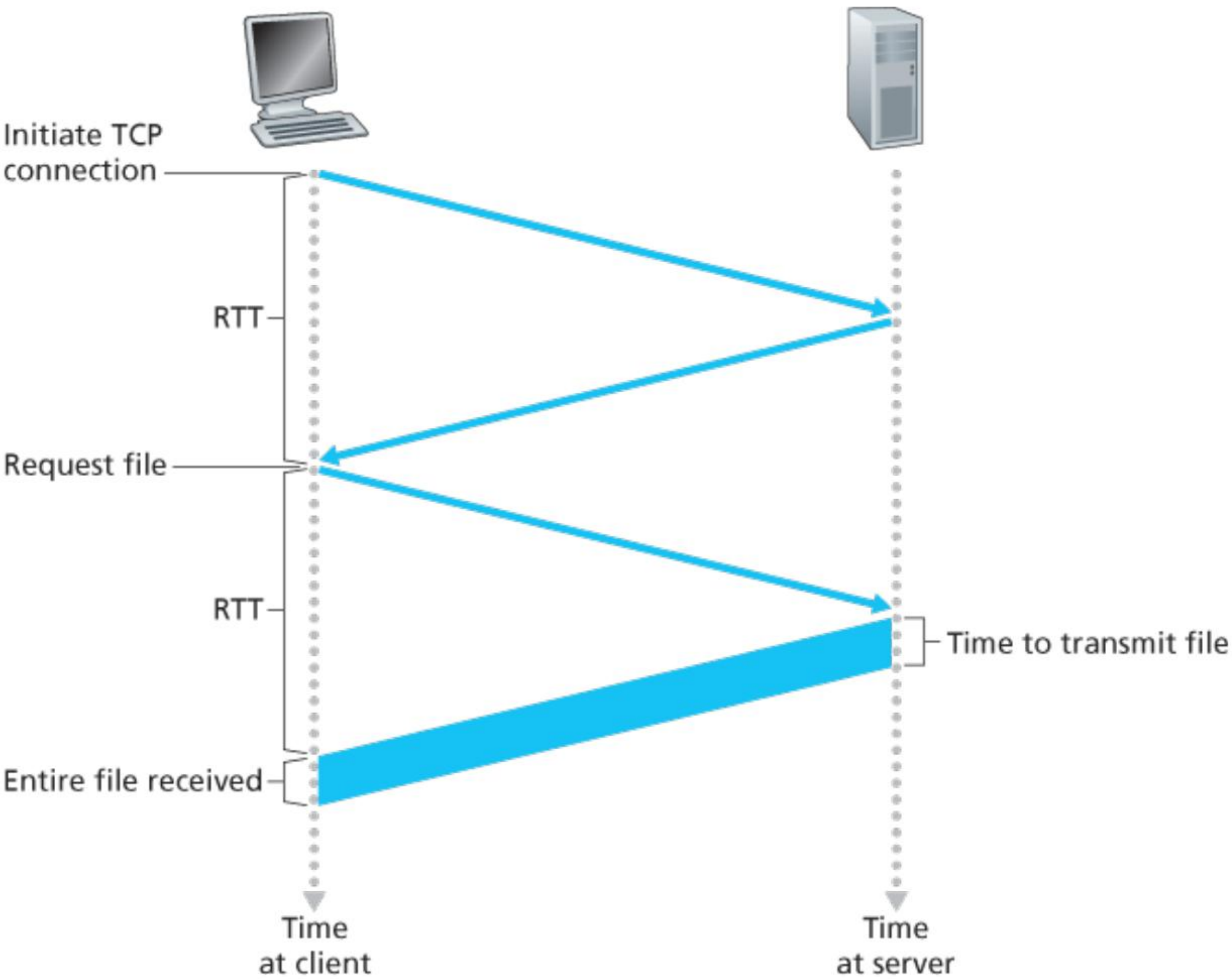
ऊपर वर्णित चरणों में, हम जानबूझकर इस बारे में अस्पष्ट थे कि ग्राहक 10 प्राप्त करता है या नहीं

10 सीरियल TCP कनेक्शन पर JPEG, या फिर कुछ JPEG समानांतर TCP कनेक्शन पर प्राप्त किए गए हैं। वास्तव में, उपयोगकर्ता आधुनिक ब्राउज़र को समानांतरता की डिग्री को नियंत्रित करने के लिए कॉन्फ़िगर कर सकते हैं। अपने डिफ़ॉल्ट मोड में, अधिकांश ब्राउज़र 5 से 10 समानांतर TCP कनेक्शन खोलते हैं, और इनमें से प्रत्येक कनेक्शन एक अनुरोध-प्रतिक्रिया लेनदेन को संभालता है। यदि उपयोगकर्ता चाहे, तो समानांतर की अधिकतम संख्या

कनेक्शन को एक पर सेट किया जा सकता है, जिस स्थिति में 10 कनेक्शन क्रमिक रूप से स्थापित किए जाते हैं। जैसा कि हम आगे देखेंगे अगले अध्याय में, समानांतर कनेक्शन का उपयोग प्रतिक्रिया समय को कम करता है।

आगे बढ़ने से पहले, आइए एक बैक-ऑफ़-द-लिफाफा गणना करें, जिससे यह अनुमान लगाया जा सके कि क्लाइंट द्वारा बेस HTML फ़ाइल का अनुरोध करने से लेकर क्लाइंट द्वारा पूरी फ़ाइल प्राप्त होने तक कितना समय बीतता है। इसके लिए, हम राउंड-ट्रिप समय (RTT) को परिभाषित करते हैं, जो एक छोटे पैकेट को यात्रा करने में लगने वाला समय है
क्लाइंट से सर्वर तक और फिर वापस क्लाइंट तक। RTT में पैकेट-प्रसार विलंब, पैकेट-
मध्यवर्ती राउटर और स्विच में कतारबद्ध देरी, और पैकेट-प्रसंस्करण देरी। (ये देरी (अनुभाग 1.4 में चर्चा की गई है ।) अब विचार करें कि जब कोई उपयोगकर्ता हाइपरलिंक पर क्लिक करता है तो क्या होता है। जैसा कि चित्र 2.7 में दिखाया गया है, यह ब्राउज़र को ब्राउज़र और वेब सर्वर के बीच एक TCP कनेक्शन आरंभ करने का कारण बनता है; इसमें एक "तीन-तरफ़ा हैंडशेक" शामिल है - क्लाइंट सर्वर को एक छोटा TCP सेगमेंट भेजता है, सर्वर एक छोटे TCP सेगमेंट के साथ स्वीकार करता है और प्रतिक्रिया करता है, और अंत में, क्लाइंट सर्वर को वापस स्वीकार करता है। तीन-तरफ़ा हैंडशेक के पहले दो भाग एक RTT लेते हैं। हैंडशेक के पहले दो भाग पूरे करने के बाद, क्लाइंट तीन-तरफ़ा हैंडशेक के तीसरे भाग (स्वीकृति) के साथ संयुक्त HTTP अनुरोध संदेश TCP कनेक्शन में भेजता है। एक बार अनुरोध

संदेश आता है



चित्र 2.7 HTML फ़ाइल का अनुरोध करने और प्राप्त करने के लिए आवश्यक समय की गणना

सर्वर, सर्वर HTML फ़ाइल को TCP कनेक्शन में भेजता है। यह HTTP अनुरोध/प्रतिक्रिया एक और RTT खा जाती है। इस प्रकार, मोटे तौर पर, कुल प्रतिक्रिया समय दो RTTs प्लस ट्रांसमिशन समय है
HTML फ़ाइल का सर्वर.

HTTP स्थायी कनेक्शन के साथ

गैर-स्थायी कनेक्शन में कुछ कमियाँ हैं। सबसे पहले, एक नया कनेक्शन होना चाहिए

प्रत्येक अनुरोधित ऑब्जेक्ट के लिए स्थापित और बनाए रखा जाता है । इनमें से प्रत्येक कनेक्शन के लिए, TCP बफ़र्स आवंटित किए जाने चाहिए और क्लाइंट और सर्वर दोनों में TCP वैरिएबल रखे जाने चाहिए। यह वेब सर्वर पर एक महत्वपूर्ण बोझ डाल सकता है, जो एक साथ सैकड़ों अलग-अलग क्लाइंट से अनुरोधों की सेवा कर सकता है। दूसरा, जैसा कि हमने अभी बताया, प्रत्येक ऑब्जेक्ट को दो RTT की डिलीवरी देरी का सामना करना पड़ता है - एक RTT TCP कनेक्शन स्थापित करने के लिए और एक RTT ऑब्जेक्ट का अनुरोध करने और प्राप्त करने के लिए।

HTTP 1.1 स्थायी कनेक्शन के साथ, सर्वर प्रतिक्रिया भेजने के बाद TCP कनेक्शन को खुला छोड़ देता है। एक ही क्लाइंट और सर्वर के बीच बाद के अनुरोध और प्रतिक्रियाएँ एक ही कनेक्शन पर भेजी जा सकती हैं। विशेष रूप से, एक संपूर्ण वेब पेज (ऊपर दिए गए उदाहरण में, बेस HTML फ़ाइल और 10 छवियाँ) को एक ही स्थायी TCP कनेक्शन पर भेजा जा सकता है। इसके अलावा, एक ही सर्वर पर रहने वाले कई वेब पेजों को सर्वर से एक ही क्लाइंट को एक ही स्थायी TCP कनेक्शन पर भेजा जा सकता है। ऑब्जेक्ट के लिए ये अनुरोध लंबित अनुरोधों (पाइपलाइनिंग) के उत्तरों की प्रतीक्षा किए बिना, बैक-टू-बैक किए जा सकते हैं। आम तौर पर, HTTP सर्वर एक कनेक्शन को बंद कर देता है जब इसका उपयोग एक निश्चित समय (एक कॉन्फ़िगर करने योग्य टाइमआउट अंतराल) के लिए नहीं किया जाता है। जब सर्वर को बैक-टू-बैक अनुरोध प्राप्त होते हैं, तो यह ऑब्जेक्ट को बैक-टू-बैक भेजता है। HTTP का डिफ़ॉल्ट मोड पाइपलाइनिंग के साथ स्थायी कनेक्शन का उपयोग करता है। अधिकांश

हाल ही में, HTTP/2 [RFC 7540] HTTP 1.1 पर आधारित है, जो एक ही कनेक्शन में कई अनुरोधों और उत्तरों को आपस में जोड़ने की अनुमति देता है, और इस कनेक्शन के भीतर HTTP संदेश अनुरोधों और उत्तरों को प्राथमिकता देने के लिए एक तंत्र प्रदान करता है। हम गैर-स्थायी और के प्रदर्शन की मात्रात्मक रूप से तुलना करेंगे

अध्याय 2 और 3 के होमवर्क समस्याओं में लगातार कनेक्शन। आपको [हेइडेमैन 1997; नीलसन 1997; आरएफसी 7540] देखने के लिए भी प्रोत्साहित किया जाता है।

2.2.3 HTTP संदेश प्रारूप

HTTP विनिर्देशों [RFC 1945; RFC 2616; RFC 7540] में HTTP संदेश प्रारूपों की परिभाषाएँ शामिल हैं। HTTP संदेश दो प्रकार के होते हैं, अनुरोध संदेश और प्रतिक्रिया संदेश,

दोनों की चर्चा नीचे की गई है।

HTTP अनुरोध संदेश

नीचे हम एक सामान्य HTTP अनुरोध संदेश प्रदान करते हैं:

```
/somedir/page.html HTTP/1.1 प्राप्त करें
होस्ट: www.someschool.edu
कनेक्शन: बंद
उपयोगकर्ता-एजेंट: मोज़िला/5.0
स्वीकार-भाषा: fr
```

इस सरल अनुरोध संदेश पर बारीकी से नज़र डालकर हम बहुत कुछ सीख सकते हैं। सबसे पहले, हम देखते हैं कि संदेश साधारण ASCII पाठ में लिखा गया है, ताकि आपका सामान्य कंप्यूटर-साक्षर व्यक्ति इसे पढ़ सके। दूसरा, हम देखते हैं कि संदेश में पाँच पंक्तियाँ हैं, जिनमें से प्रत्येक के बाद एक कैरिज रिटर्न और एक लाइन फ़ीड है। अंतिम पंक्ति के बाद एक अतिरिक्त कैरिज रिटर्न और लाइन फ़ीड है। हालाँकि इस विशेष अनुरोध संदेश में पाँच पंक्तियाँ हैं, एक अनुरोध संदेश में कई और पंक्तियाँ या एक पंक्ति जितनी कम भी हो सकती हैं।

HTTP अनुरोध संदेश की पहली पंक्ति को अनुरोध पंक्ति कहा जाता है; बाद की पंक्तियों को हेडर पंक्तियाँ कहा जाता है। अनुरोध पंक्ति में तीन फ़ील्ड होते हैं: विधि फ़ील्ड, URL फ़ील्ड और HTTP

संस्करण फ़ील्ड। विधि फ़ील्ड कई अलग-अलग मान ले सकती है, जिसमें GET, POST, HEAD,

PUT, और DELETE . HTTP अनुरोध संदेशों का बड़ा हिस्सा GET विधि का उपयोग करता है। GET विधि का उपयोग तब किया जाता है जब ब्राउज़र किसी ऑब्जेक्ट का अनुरोध करता है, जिसमें अनुरोधित ऑब्जेक्ट को URL में पहचाना जाता है

फ़ील्ड। इस उदाहरण में, ब्राउज़र ऑब्जेक्ट /somedir/page.html **का अनुरोध कर रहा है** । संस्करण स्व-व्याख्यात्मक है; इस उदाहरण में, ब्राउज़र संस्करण HTTP/1.1 को लागू करता है।

अब आइए उदाहरण में हेडर लाइनों को देखें। हेडर लाइन Host: www.someschool.edu उस होस्ट को निर्दिष्ट करती है जिस पर ऑब्जेक्ट रहता है। आपको लग सकता है कि यह हेडर लाइन अनावश्यक है, क्योंकि

होस्ट के लिए पहले से ही एक TCP कनेक्शन मौजूद है। लेकिन, जैसा कि हम अनुभाग 2.2.5 में देखेंगे, जानकारी

होस्ट हेडर लाइन द्वारा प्रदान किया गया कनेक्शन वेब प्रॉक्सी कैश के लिए आवश्यक है। कनेक्शन को शामिल करके:

हेडर लाइन को बंद करके , ब्राउज़र सर्वर को बता रहा है कि वह लगातार परेशानी से परेशान नहीं होना चाहता है

कनेक्शन; यह चाहता है कि सर्वर अनुरोधित ऑब्जेक्ट भेजने के बाद कनेक्शन बंद कर दे ।

एजेंट: हेडर लाइन उपयोगकर्ता एजेंट को निर्दिष्ट करती है, अर्थात, वह ब्राउज़र प्रकार जो सर्वर से अनुरोध कर रहा है। यहाँ उपयोगकर्ता एजेंट Mozilla/5.0 है, जो फ़ायरफ़ॉक्स ब्राउज़र है। यह हेडर लाइन उपयोगी है क्योंकि सर्वर वास्तव में एक ही ऑब्जेक्ट के विभिन्न संस्करणों को विभिन्न प्रकार के उपयोगकर्ता एजेंटों को भेज सकता है। (प्रत्येक

संस्करण एक ही यूआरएल द्वारा संबोधित है।) अंत में, Accept-language: हेडर इंगित करता है कि उपयोगकर्ता ऑब्जेक्ट का फ़्रेंच संस्करण प्राप्त करना पसंद करता है, यदि ऐसा ऑब्जेक्ट सर्वर पर मौजूद है;

अन्यथा, सर्वर को अपना डिफ़ॉल्ट संस्करण भेजना चाहिए। Accept-language: हेडर HTTP में उपलब्ध कई कंटेंट नेगोशिएशन हेडर में से एक है।

एक उदाहरण देखने के बाद, आइए अब अनुरोध संदेश के सामान्य प्रारूप को देखें, जैसा कि नीचे दिखाया गया है।

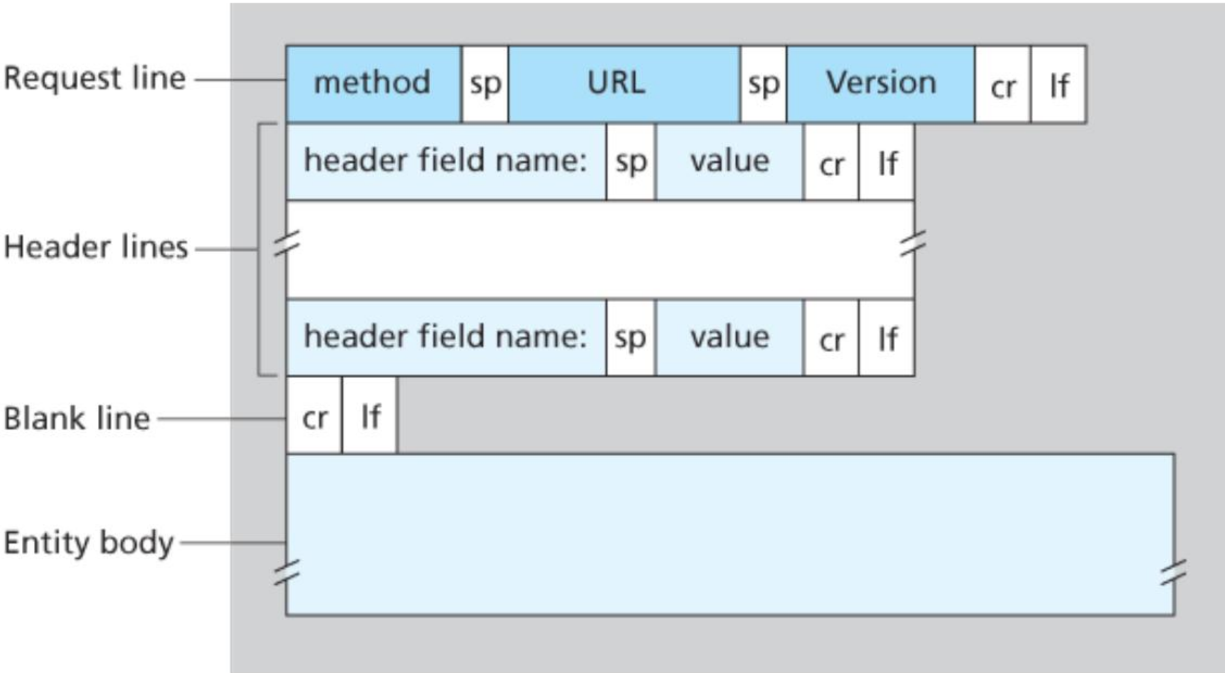
चित्र 2.8. हम देखते हैं कि सामान्य प्रारूप हमारे पिछले उदाहरण का बहुत करीब से अनुसरण करता है। आपने देखा होगा,

हालाँकि, हेडर लाइनों (और अतिरिक्त कैरिज रिटर्न और लाइन फीड) के बाद एक "इकाई" है

बॉडी।" GET **विधि** के साथ इकाई बॉडी खाली होती है , लेकिन POST विधि के साथ इसका उपयोग किया जाता है। HTTP

क्लाइंट अक्सर POST विधि का उपयोग करता है जब उपयोगकर्ता एक फ़ॉर्म भरता है - उदाहरण के लिए, जब कोई उपयोगकर्ता प्रदान करता है

खोज इंजन में खोज शब्द भेजना। POST संदेश के साथ, उपयोगकर्ता अभी भी सर्वर से वेब पेज का अनुरोध कर रहा है, लेकिन वेब पेज की विशिष्ट सामग्री



चित्र 2.8 HTTP अनुरोध संदेश का सामान्य प्रारूप

इस पर निर्भर करता है कि उपयोगकर्ता ने फ़ॉर्म फ़ील्ड में क्या दर्ज किया है। यदि विधि फ़ील्ड का मान POST **है** तो इकाई निकाय में वह शामिल है जो उपयोगकर्ता ने , तो फिर फ़ॉर्म फ़ील्ड में दर्ज किया है।

हम लापरवाह होंगे यदि हम यह उल्लेख न करें कि फॉर्म के साथ उत्पन्न अनुरोध आवश्यक रूप से इसका उपयोग नहीं करता है

POST विधि। इसके बजाय, HTML फॉर्म अक्सर GET **विधि** का **उपयोग करते हैं** और इनपुट किए गए डेटा (इन

अनुरोधित URL में फ़ॉर्म फ़ील्ड) दर्ज करें। उदाहरण के लिए, यदि कोई फ़ॉर्म GET विधि का उपयोग करता है, तो उसके दो फ़ील्ड हैं, और

दोनों क्षेत्रों के इनपुट बंदर और केले **हैं** , तो URL की संरचना इस प्रकार होगी

www.somesite.com/animalsearch?monkeys&bananas अपने दैनिक वेब सर्फिंग में, आपने संभवतः इस प्रकार के विस्तारित URL देखे होंगे।

HEAD **विधि** GET **विधि** के समान है । **जब सर्वर** HEAD **विधि** से अनुरोध प्राप्त करता है , तो वह HTTP संदेश के साथ प्रतिक्रिया करता है, लेकिन अनुरोधित ऑब्जेक्ट को छोड़ देता है ।

डेवलपर्स अक्सर डीबगिंग के लिए HEAD **विधि का उपयोग करते हैं**। PUT विधि का उपयोग अक्सर वेब प्रकाशन उपकरणों के साथ किया जाता है। यह उपयोगकर्ता को किसी विशिष्ट पथ (निर्देशिका) पर किसी ऑब्जेक्ट को अपलोड करने की अनुमति देता है

वेब सर्वर। PUT विधि का उपयोग उन अनुप्रयोगों द्वारा भी किया जाता है जिन्हें वेब सर्वर पर ऑब्जेक्ट अपलोड करने की आवश्यकता होती है।

DELETE विधि किसी उपयोगकर्ता या अनुप्रयोग को वेब सर्वर पर किसी ऑब्जेक्ट को हटाने की अनुमति देती है।

HTTP प्रतिक्रिया संदेश

नीचे हम एक सामान्य HTTP प्रतिक्रिया संदेश प्रदान करते हैं। यह प्रतिक्रिया संदेश अभी चर्चा किए गए उदाहरण अनुरोध संदेश का जवाब हो सकता है।

```
HTTP/1.1 200 ठीक
कनेक्शन: बंद
दिनांक: मंगल, 18 अगस्त 2015 15:44:04 GMT
सर्वर: अपाचे/2.2.3 (CentOS)
अंतिम संशोधन: मंगल, 18 अगस्त 2015 15:11:03 GMT
सामग्री-लंबाई: 6821
सामग्री-प्रकार: text/html

(डेटा डेटा डेटा डेटा डेटा ...)
```

आइए इस प्रतिक्रिया संदेश पर ध्यान से नज़र डालें। इसमें तीन खंड हैं: एक आरंभिक स्थिति पंक्ति, छह हेडर पंक्तियाँ, और फिर इकाई निकाय। इकाई निकाय संदेश का मुख्य भाग है - इसमें शामिल है

अनुरोधित ऑब्जेक्ट स्वयं (डेटा डेटा डेटा डेटा डेटा ... द्वारा दर्शाया गया)। स्थिति रेखा में तीन फ़ील्ड हैं: प्रोटोकॉल संस्करण फ़ील्ड, एक स्थिति कोड, और एक संबंधित स्थिति संदेश। इस उदाहरण में, स्थिति रेखा इंगित करती है कि सर्वर HTTP/1.1 का उपयोग कर रहा है और सब कुछ ठीक है (अर्थात, सर्वर ने अनुरोधित ऑब्जेक्ट ढूंढ लिया है, और भेज रहा है)।

अब हेडर लाइन्स पर नज़र डालते हैं। सर्वर Connection: close हेडर लाइन का इस्तेमाल करके बताता है कि सर्वर को क्या करना है।

क्लाइंट को यह संदेश भेजने के बाद TCP कनेक्शन बंद करने के लिए कहा जाता है। Date: हेडर लाइन उस समय और दिनांक को इंगित करती है जब HTTP प्रतिक्रिया सर्वर द्वारा बनाई और भेजी गई थी। ध्यान दें कि यह वह समय नहीं है जब ऑब्जेक्ट बनाया गया था या अंतिम बार संशोधित किया गया था; यह वह समय है जब सर्वर अपने फ़ाइल सिस्टम से ऑब्जेक्ट को पुनर्प्राप्त करता है, ऑब्जेक्ट को प्रतिक्रिया संदेश में सम्मिलित करता है, और प्रतिक्रिया भेजता है

संदेश। सर्वर: हेडर लाइन यह इंगित करती है कि संदेश अपाचे वेब द्वारा उत्पन्न किया गया था

सर्वर; यह HTTP अनुरोध संदेश में User-agent: हेडर लाइन के समान है ।

संशोधित: हेडर लाइन उस समय और तारीख को इंगित करती है जब ऑब्जेक्ट बनाया गया था या अंतिम बार संशोधित किया गया था।

Last-Modified: हेडर, जिसके बारे में हम जल्द ही विस्तार से चर्चा करेंगे, ऑब्जेक्ट कैशिंग के लिए महत्वपूर्ण है, दोनों में

स्थानीय क्लाइंट और नेटवर्क कैश सर्वर (जिसे प्रॉक्सी सर्वर भी कहा जाता है) में। सामग्री-लंबाई:

हेडर लाइन भेजी जा रही वस्तु में बाइट्स की संख्या को इंगित करती है। Content-Type: हेडर लाइन यह इंगित करती है कि इकाई बॉडी में वस्तु HTML टेक्स्ट है। (ऑब्जेक्ट प्रकार आधिकारिक तौर पर द्वारा इंगित किया जाता है

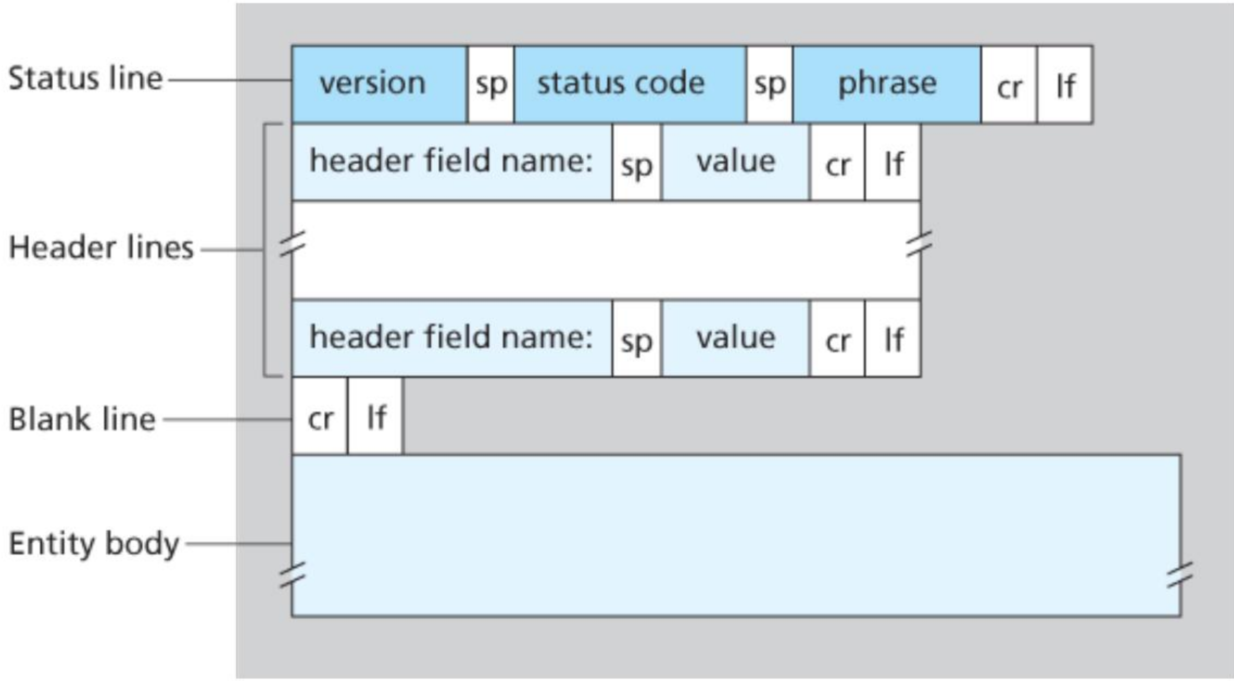
Content-Type: हेडर द्वारा और फ़ाइल एक्सटेंशन द्वारा नहीं।)

एक उदाहरण देखने के बाद, आइए अब प्रतिक्रिया संदेश के सामान्य प्रारूप की जांच करें, जो है

चित्र 2.9 में दिखाया गया है । प्रतिक्रिया संदेश का यह सामान्य प्रारूप प्रतिक्रिया संदेश के पिछले उदाहरण से मेल खाता है। आइए स्टेटस कोड और उनके वाक्यांशों के बारे में कुछ अतिरिक्त शब्द कहें। स्टेटस

कोड और संबंधित वाक्यांश अनुरोध के परिणाम को इंगित करते हैं। कुछ सामान्य स्थिति कोड और संबंधित वाक्यांशों में शामिल हैं:

- 200 OK: अनुरोध सफल हुआ और सूचना प्रतिक्रिया में वापस आ गई।
- 301 स्थायी रूप से स्थानांतरित: अनुरोधित ऑब्जेक्ट स्थायी रूप से स्थानांतरित कर दिया गया है; नया यूआरएल है स्थान में निर्दिष्ट : प्रतिक्रिया संदेश का शीर्षलेख। क्लाइंट सॉफ़्टवेयर स्वचालित रूप से नया URL पुनः प्राप्त करें.
- 400 खराब अनुरोध: यह एक सामान्य त्रुटि कोड है जो यह दर्शाता है कि अनुरोध सर्वर द्वारा समझा नहीं जा सका।



चित्र 2.9 HTTP प्रतिक्रिया संदेश का सामान्य प्रारूप

- 404 नहीं मिला: अनुरोधित दस्तावेज़ इस सर्वर पर मौजूद नहीं है।
- 505 HTTP संस्करण समर्थित नहीं है: अनुरोधित HTTP प्रोटोकॉल संस्करण समर्थित नहीं है सर्वर द्वारा.

आप वास्तविक HTTP प्रतिक्रिया संदेश कैसे देखना चाहेंगे? यह अत्यधिक अनुशंसित है और करना बहुत आसान है! सबसे पहले अपने पसंदीदा वेब सर्वर में टेलनेट करें। फिर सर्वर पर रखे किसी ऑब्जेक्ट के लिए एक-लाइन अनुरोध संदेश टाइप करें। उदाहरण के लिए, यदि आपके पास कमांड प्रॉम्प्ट तक पहुँच है, तो टाइप करें:

VideoNote

HTTP प्रोटोकॉल की जांच के लिए Wireshark का उपयोग करना

```
टेलनेट gaia.cs.umass.edu 80

/kurose_ross/interactive/index.php HTTP/1.1 प्राप्त करें
होस्ट: gaia.cs.umass.edu
```

(अंतिम पंक्ति टाइप करने के बाद कैरिज रिटर्न को दो बार दबाएँ।) यह पोर्ट 80 पर एक TCP कनेक्शन खोलता है।

होस्ट gaia.cs.umass.edu और फिर HTTP अनुरोध संदेश भेजता है। आपको एक प्रतिक्रिया संदेश देखना चाहिए जिसमें इस पाठ्यपुस्तक के लिए इंटरैक्टिव होमवर्क समस्याओं के लिए आधार HTML फ़ाइल शामिल है। अगर

यदि आप केवल HTTP संदेश पंक्तियां देखना चाहते हैं और ऑब्जेक्ट प्राप्त नहीं करना चाहते हैं, तो GET को HEAD से **प्रतिस्थापित करें** ।

इस अनुभाग में हमने कई हेडर लाइनों पर चर्चा की है जिनका उपयोग HTTP अनुरोध और प्रतिक्रिया संदेशों में किया जा सकता है। HTTP विनिर्देश कई, कई और हेडर लाइनों को परिभाषित करता है जिन्हें ब्राउज़र, वेब सर्वर और नेटवर्क कैश सर्वर द्वारा डाला जा सकता है। हमने हेडर लाइनों की कुल संख्या में से केवल कुछ को कवर किया है। हम नीचे कुछ और कवर करेंगे और जब हम चर्चा करेंगे तो एक और छोटी संख्या को कवर करेंगे

नेटवर्क वेब कैशिंग अनुभाग 2.2.5 में। HTTP की एक अत्यंत पठनीय और व्यापक चर्चा
प्रोटोकॉल, इसके हेडर और स्थिति कोड सहित, [कृष्णमूर्ति 2001] में दिया गया है।

ब्राउज़र कैसे तय करता है कि अनुरोध संदेश में कौन सी हेडर लाइनें शामिल की जाएँ? वेब सर्वर कैसे तय करता है कि प्रतिक्रिया संदेश में कौन सी हेडर लाइनें शामिल की जाएँ? ब्राउज़र ब्राउज़र के प्रकार और संस्करण (उदाहरण के लिए, HTTP/1.0 ब्राउज़र कोई 1.1 हेडर लाइन नहीं बनाएगा), ब्राउज़र के उपयोगकर्ता कॉन्फ़िगरेशन (उदाहरण के लिए, पसंदीदा भाषा) और ब्राउज़र में वर्तमान में ऑब्जेक्ट का कैश्ड, लेकिन संभवतः पुराना संस्करण है या नहीं, के आधार पर हेडर लाइनें उत्पन्न करेगा। वेब सर्वर इसी तरह व्यवहार करते हैं: अलग-अलग उत्पाद, संस्करण और कॉन्फ़िगरेशन हैं, जो सभी प्रभावित करते हैं कि प्रतिक्रिया संदेशों में कौन सी हेडर लाइनें शामिल की जाएँ।

2.2.4 उपयोगकर्ता-सर्वर इंटरैक्शन: कुकीज़

हमने ऊपर बताया कि HTTP सर्वर स्टेटलेस होता है। इससे सर्वर डिज़ाइन सरल हो जाता है और इंजीनियरों को उच्च-प्रदर्शन वाले वेब सर्वर विकसित करने की अनुमति मिल जाती है जो हज़ारों TCP कनेक्शन को एक साथ संभाल सकते हैं। हालाँकि, किसी वेब साइट के लिए उपयोगकर्ताओं की पहचान करना अक्सर वांछनीय होता है, या तो इसलिए क्योंकि सर्वर उपयोगकर्ता की पहुँच को प्रतिबंधित करना चाहता है या इसलिए क्योंकि वह उपयोगकर्ता की पहचान के आधार पर सामग्री परोसना चाहता है।

इन उद्देश्यों के लिए, HTTP कुकीज़ का उपयोग करता है। [RFC 6265] में परिभाषित कुकीज़ , साइटों को उपयोगकर्ताओं पर नज़र रखने की अनुमति देती हैं।
आज अधिकांश प्रमुख वाणिज्यिक वेब साइटें कुकीज़ का उपयोग करती हैं।

जैसा कि चित्र 2.10 में दिखाया गया है , कुकी तकनीक के चार घटक हैं: (1) HTTP प्रतिक्रिया संदेश में कुकी हेडर लाइन; (2) HTTP अनुरोध संदेश में कुकी हेडर लाइन; (3) HTTP सर्वर पर रखी गई कुकी फ़ाइल।

उपयोगकर्ता के अंतिम सिस्टम पर आधारित और उपयोगकर्ता के ब्राउज़र द्वारा प्रबंधित; और (4) वेब साइट पर एक बैक-एंड डेटाबेस।

चित्र 2.10 का उपयोग करके , आइए कुकीज़ कैसे काम करती हैं, इसका एक उदाहरण देखें। मान लीजिए सुसान, जो हमेशा अपने घर के पीसी से इंटरनेट एक्सप्लोरर का उपयोग करके वेब एक्सेस करती है, Amazon.com **से संपर्क करती है** पहली बार के लिए।

मान लीजिए कि वह पहले भी eBay साइट पर जा चुकी है। जब अनुरोध आता है

अमेज़न वेब सर्वर, सर्वर एक अद्वितीय पहचान संख्या बनाता है और अपने बैक-एंड डेटाबेस में एक प्रविष्टि बनाता है जिसे पहचान संख्या द्वारा अनुक्रमित किया जाता है। फिर अमेज़न वेब सर्वर प्रतिक्रिया देता है

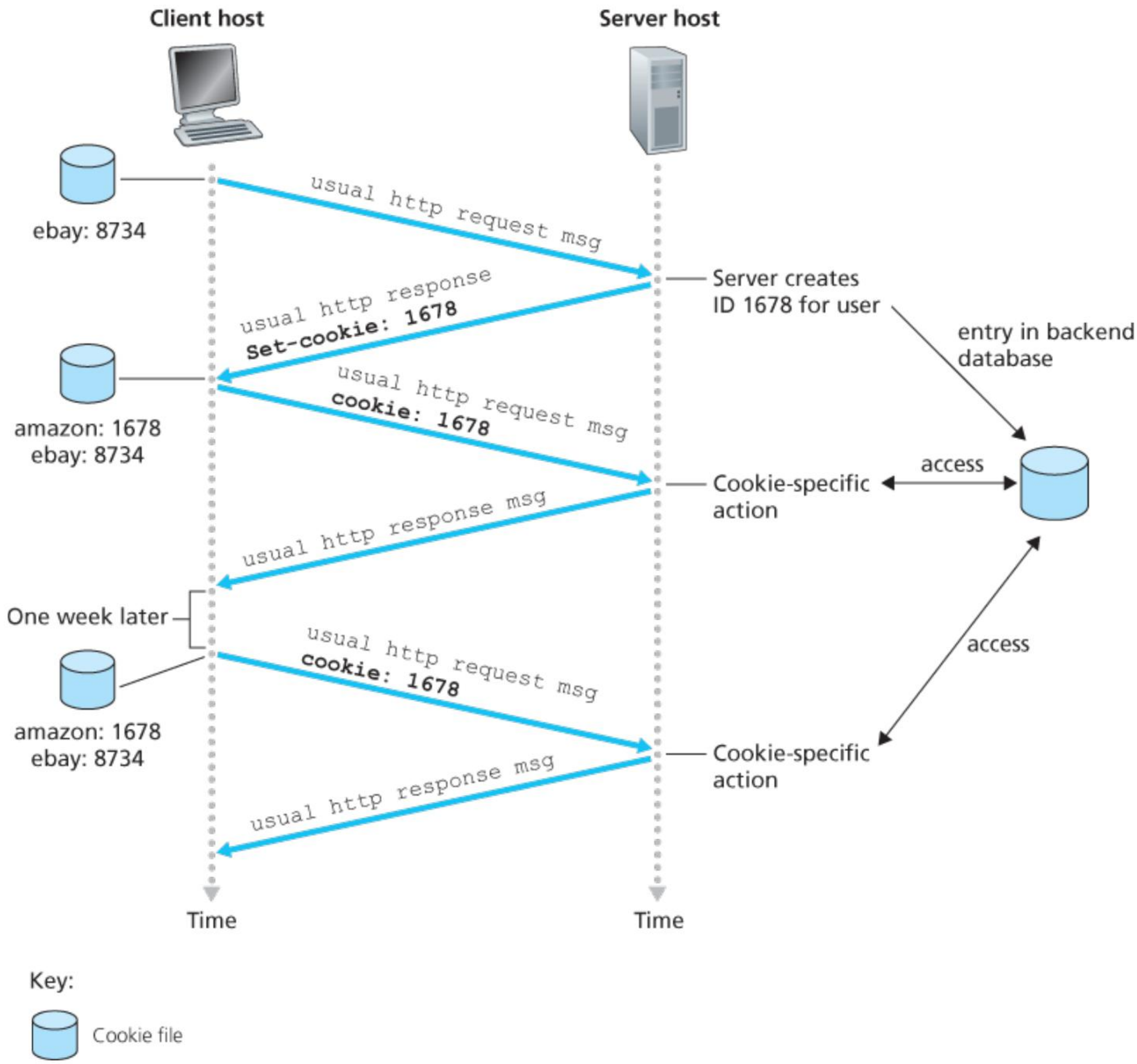
सुसान के ब्राउज़र में, HTTP प्रतिक्रिया में Set-cookie: हेडर शामिल है, जिसमें पहचान संख्या शामिल है। उदाहरण के लिए, हेडर लाइन हो सकती है:

```
सेट-कुकी: 1678
```

जब सुसान के ब्राउज़र को HTTP प्रतिक्रिया संदेश प्राप्त होता है, तो उसे Set-cookie: हेडर दिखाई देता है। फिर ब्राउज़र उस विशेष कुकी फ़ाइल में एक पंक्ति जोड़ता है जिसे वह प्रबंधित करता है। इस पंक्ति में होस्टनाम शामिल है

सर्वर का और Set-cookie: हेडर में पहचान संख्या। ध्यान दें कि कुकी फ़ाइल

ईबे के लिए पहले से ही एक प्रविष्टि है, क्योंकि सुसान ने अतीत में उस साइट का दौरा किया है। जैसा कि सुसान अमेज़ॅन साइट ब्राउज़ करना जारी रखती है, हर बार जब वह एक वेब पेज का अनुरोध करती है, तो उसका ब्राउज़र उसकी कुकी फ़ाइल से परामर्श करता है, इस साइट के लिए उसकी पहचान संख्या निकालता है, और एक कुकी हेडर लाइन डालता है जो



चित्र 2.10 कुकीज़ के साथ उपयोगकर्ता स्थिति बनाए रखना

HTTP अनुरोध में पहचान संख्या शामिल है। विशेष रूप से, उसके प्रत्येक HTTP अनुरोध में अमेज़न सर्वर में हेडर लाइन शामिल है:

```
कुकी: 1678
```

इस तरह, अमेज़न सर्वर अमेज़न साइट पर सुसान की गतिविधि को ट्रैक करने में सक्षम है। हालाँकि अमेज़न वेब साइट को सुसान का नाम नहीं पता है, लेकिन यह ठीक से जानता है कि उपयोगकर्ता 1678 किन पृष्ठों पर है।

किस क्रम में और किस समय पर दौरा किया गया! Amazon अपनी शॉपिंग कार्ट सेवा प्रदान करने के लिए कुकीज़ का उपयोग करता है—

अमेज़न सुसान की सभी इच्छित खरीदारी की सूची बनाए रख सकता है, ताकि वह उनका भुगतान कर सके

सत्र के अंत में सामूहिक रूप से।

मान लीजिए, एक सप्ताह बाद यदि सुसान अमेज़न की साइट पर वापस आती है, तो उसका ब्राउज़र हेडर लाइन डालना जारी रखेगा

कुकी: 1678 अनुरोध संदेशों में। Amazon सुसान को उन वेब पेजों के आधार पर भी उत्पादों की सिफारिश करता है, जो उसने पहले Amazon पर देखे हैं। अगर सुसान खुद को Amazon पर रजिस्टर भी करती है—पूरा नाम, ईमेल पता, डाक पता और क्रेडिट कार्ड की जानकारी देती है—तो Amazon इस जानकारी को अपने डेटाबेस में शामिल कर सकता है, जिससे सुसान का नाम उसकी पहचान संख्या (और साइट पर उसके द्वारा पहले देखे गए सभी पेज!) से जुड़ जाएगा। इस तरह से Amazon और अन्य ई-कॉमर्स साइट्स "वन-क्लिक शॉपिंग" प्रदान करती हैं—जब सुसान किसी बाद की यात्रा के दौरान कोई आइटम खरीदना चुनती है,

उसे अपना नाम, क्रेडिट कार्ड नंबर या पता दोबारा दर्ज करने की आवश्यकता नहीं है।

इस चर्चा से हम देखते हैं कि कुकीज़ का उपयोग उपयोगकर्ता की पहचान करने के लिए किया जा सकता है। जब कोई उपयोगकर्ता पहली बार किसी साइट पर जाता है, तो उपयोगकर्ता एक उपयोगकर्ता पहचान (संभवतः उसका नाम) प्रदान कर सकता है। बाद के सत्रों के दौरान, ब्राउज़र सर्वर को कुकी हेडर भेजता है, जिससे सर्वर को उपयोगकर्ता की पहचान हो जाती है।
इस प्रकार कुकीज़ का उपयोग स्टेटलेस HTTP के शीर्ष पर उपयोगकर्ता सत्र परत बनाने के लिए किया जा सकता है। उदाहरण के लिए, जब कोई उपयोगकर्ता वेब-आधारित ई-मेल एप्लिकेशन (जैसे हॉटमेल) में लॉग इन करता है, तो ब्राउज़र सर्वर को कुकी जानकारी भेजता है, जिससे सर्वर को एप्लिकेशन के साथ उपयोगकर्ता के सत्र के दौरान उपयोगकर्ता की पहचान करने की अनुमति मिलती है।

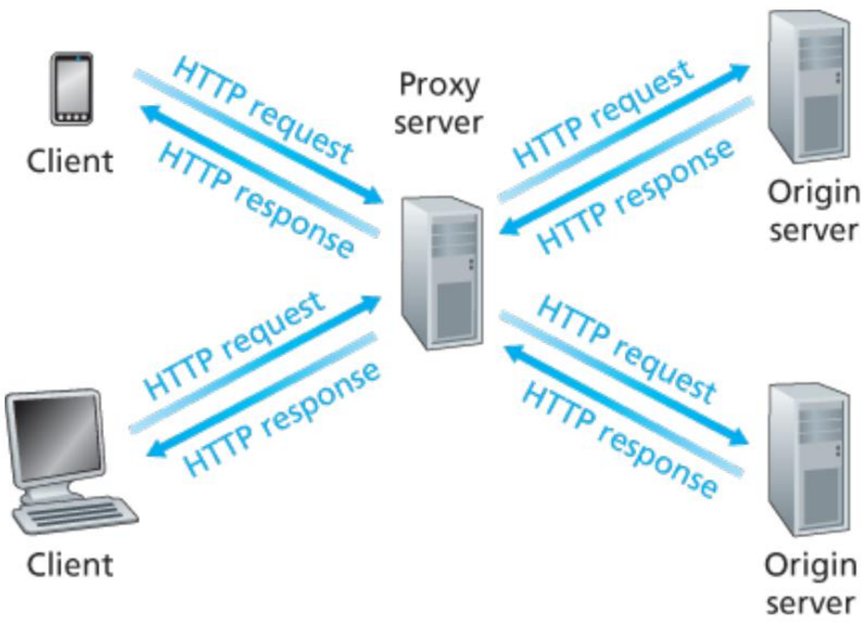
हालाँकि कुकीज़ अक्सर उपयोगकर्ता के लिए इंटरनेट शॉपिंग अनुभव को सरल बनाती हैं, लेकिन वे विवादास्पद हैं क्योंकि उन्हें गोपनीयता का उल्लंघन भी माना जा सकता है। जैसा कि हमने अभी देखा, कुकीज़ और उपयोगकर्ता द्वारा दी गई खाता जानकारी के संयोजन का उपयोग करके, एक वेब साइट उपयोगकर्ता के बारे में बहुत कुछ जान सकती है और संभावित रूप से
इस जानकारी को किसी तीसरे पक्ष को न बेचें। कुकी सेंट्रल [कुकी सेंट्रल 2016] में कुकी विवाद पर व्यापक जानकारी शामिल है।

2.2.5 वेब कैशिंग

वेब कैश—जिसे प्रॉक्सी सर्वर भी कहा जाता है—एक नेटवर्क इकाई है **जो मूल वेब सर्वर की ओर से HTTP अनुरोधों को संतुष्ट करती है। वेब कैश** का अपना डिस्क स्टोरेज होता है और हाल ही में किए गए डेटा की प्रतियां रखता है
इस स्टोरेज में अनुरोधित ऑब्जेक्ट्स। जैसा कि चित्र 2.11 में दिखाया गया है, उपयोगकर्ता के ब्राउज़र को इस तरह से कॉन्फ़िगर किया जा सकता है कि उपयोगकर्ता के सभी HTTP अनुरोध पहले वेब कैश को निर्देशित किए जाएं। एक बार ब्राउज़र कॉन्फ़िगर हो जाने के बाद, ऑब्जेक्ट के लिए प्रत्येक ब्राउज़र अनुरोध पहले वेब कैश को निर्देशित किया जाता है। उदाहरण के लिए, मान लीजिए कि ब्राउज़र है
ऑब्जेक्ट http://www.someschool.edu/campus.gif **का अनुरोध करना** । यहाँ बताया गया है कि क्या होता है:

1. ब्राउज़र वेब कैश से एक TCP कनेक्शन स्थापित करता है और वेब कैश को ऑब्जेक्ट के लिए HTTP अनुरोध भेजता है।

2. वेब कैश यह जाँचता है कि क्या उसके पास स्थानीय रूप से संग्रहीत ऑब्जेक्ट की एक प्रति है। यदि ऐसा है, तो वेब कैश क्लाइंट ब्राउज़र को HTTP प्रतिक्रिया संदेश के माध्यम से ऑब्जेक्ट लौटाता है।



चित्र 2.11 वेब कैश के माध्यम से ऑब्जेक्ट का अनुरोध करने वाले क्लाइंट

3. यदि वेब कैश में ऑब्जेक्ट नहीं है, तो वेब कैश मूल सर्वर, यानी www.someschool.edu **से एक TCP कनेक्शन खोलता है** । फिर वेब कैश कैश-टू-सर्वर TCP कनेक्शन में ऑब्जेक्ट के लिए HTTP अनुरोध भेजता है। इस अनुरोध को प्राप्त करने के बाद, मूल सर्वर

HTTP प्रतिक्रिया के भीतर ऑब्जेक्ट को वेब कैश में भेजता है।

4. जब वेब कैश ऑब्जेक्ट प्राप्त करता है, तो वह उसकी एक प्रतिलिपि अपने स्थानीय स्टोरेज में संग्रहीत करता है और एक प्रतिलिपि, HTTP प्रतिक्रिया संदेश के माध्यम से, क्लाइंट ब्राउज़र को भेजता है (क्लाइंट ब्राउज़र और वेब कैश के बीच मौजूदा TCP कनेक्शन पर)।

ध्यान दें कि कैश एक ही समय में सर्वर और क्लाइंट दोनों होता है। जब यह किसी ब्राउज़र से अनुरोध प्राप्त करता है और प्रतिक्रियाएँ भेजता है, तो यह एक सर्वर होता है। जब यह किसी ब्राउज़र से अनुरोध भेजता है और प्रतिक्रियाएँ प्राप्त करता है, तो यह एक सर्वर होता है।
मूल सर्वर, यह एक ग्राहक है.

आम तौर पर एक वेब कैश एक ISP द्वारा खरीदा और स्थापित किया जाता है। उदाहरण के लिए, एक विश्वविद्यालय अपने कैंपस नेटवर्क पर एक कैश स्थापित कर सकता है और सभी कैंपस ब्राउज़र को कैश की ओर इंगित करने के लिए कॉन्फ़िगर कर सकता है। या एक प्रमुख आवासीय ISP (जैसे कॉमकास्ट) अपने नेटवर्क में एक या अधिक कैश स्थापित कर सकता है और अपने कैश को पहले से कॉन्फ़िगर कर सकता है।
ब्राउज़रों को स्थापित कैश की ओर इंगित करने के लिए भेजा गया।

वेब कैशिंग को इंटरनेट में दो कारणों से इस्तेमाल किया गया है। सबसे पहले, वेब कैश क्लाइंट अनुरोध के लिए प्रतिक्रिया समय को काफी हद तक कम कर सकता है, खासकर अगर क्लाइंट और मूल सर्वर के बीच की बाधा बैंडविड्थ क्लाइंट और कैश के बीच की बाधा बैंडविड्थ से बहुत कम है। अगर क्लाइंट और कैश के बीच हाई-स्पीड कनेक्शन है, जैसा कि अक्सर होता है, और अगर कैश में अनुरोधित ऑब्जेक्ट है, तो कैश क्लाइंट को ऑब्जेक्ट को तेज़ी से डिलीवर करने में सक्षम होगा।
दूसरा, जैसा कि हम जल्द ही एक उदाहरण के साथ स्पष्ट करेंगे, वेब कैश इंटरनेट पर किसी संस्थान के एक्सेस लिंक पर ट्रैफ़िक को काफी हद तक कम कर सकता है। ट्रैफ़िक को कम करके, संस्थान (उदाहरण के लिए, एक कंपनी या विश्वविद्यालय)

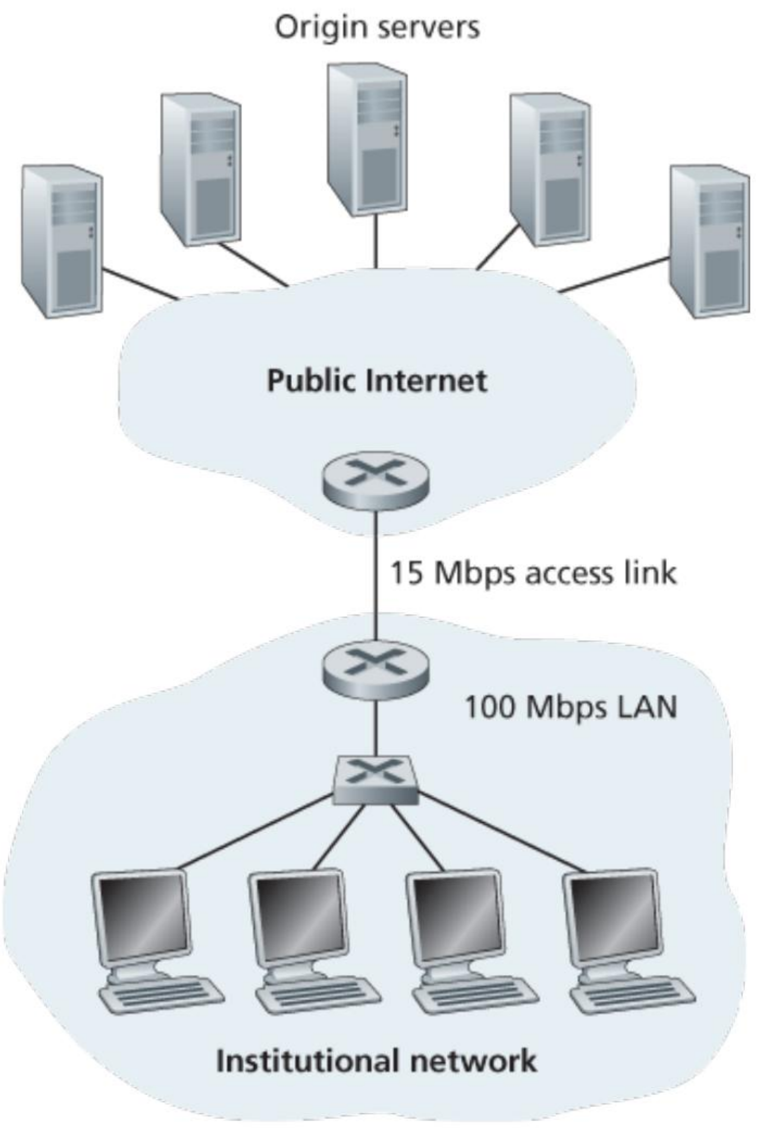
बैंडविड्थ को जल्दी से अपग्रेड नहीं करना पड़ता, जिससे लागत कम हो जाती है। इसके अलावा, वेब कैश

इंटरनेट पर वेब ट्रैफिक को काफी हद तक कम करना, जिससे सभी के लिए प्रदर्शन में सुधार हो
अनुप्रयोग.

कैश के लाभों की गहरी समझ हासिल करने के लिए, आइए एक उदाहरण पर विचार करें।

चित्र 2.12. यह चित्र दो नेटवर्क दिखाता है- संस्थागत नेटवर्क और बाकी सार्वजनिक इंटरनेट। संस्थागत नेटवर्क एक हाई-स्पीड LAN है। संस्थागत नेटवर्क में एक राउटर और एक राउटर
इंटरनेट में 15 एमबीपीएस लिंक से जुड़े हुए हैं। मूल सर्वर इंटरनेट से जुड़े हुए हैं लेकिन
दुनिया भर में स्थित है। मान लीजिए कि औसत ऑब्जेक्ट का आकार 1 Mbits है और संस्थान के ब्राउज़र से मूल सर्वर तक औसत अनुरोध दर 15 अनुरोध प्रति सेकंड है। मान लीजिए कि HTTP अनुरोध संदेश नगण्य रूप से छोटे हैं और इस प्रकार नेटवर्क या एक्सेस लिंक (संस्थागत राउटर से इंटरनेट राउटर तक) में कोई ट्रैफ़िक नहीं बनाते हैं। यह भी मान लीजिए कि जब से शुरू होता है तब से लेकर अंत तक कितना समय लगता है

चित्र 2.12 में एक्सेस लिंक के इंटरनेट साइड पर राउटर HTTP अनुरोध (IP डेटाग्राम के भीतर) को तब तक अग्रेषित करता है जब तक कि उसे प्रतिक्रिया नहीं मिल जाती (आमतौर पर कई IP डेटाग्राम के भीतर) औसतन दो सेकंड लगते हैं। अनौपचारिक रूप से, हम इस अंतिम देरी को "इंटरनेट देरी" कहते हैं।



चित्र 2.12 संस्थागत नेटवर्क और इंटरनेट के बीच अड़चन

कुल प्रतिक्रिया समय - अर्थात, ब्राउज़र द्वारा किसी ऑब्जेक्ट के अनुरोध से लेकर ऑब्जेक्ट की प्राप्ति तक का समय - LAN विलंब, पहुँच विलंब (अर्थात, दो राउटरों के बीच विलंब) और का योग है।

इंटरनेट में देरी। आइए अब इस देरी का अनुमान लगाने के लिए एक बहुत ही कच्चा हिसाब लगाते हैं।

LAN (अनुभाग 1.4.2 देखें)

(15 अनुरोध/सेकंड)⯑(1 एमबिट्स/अनुरोध)/(100 एमबीपीएस)=0.15

जबकि एक्सेस लिंक पर ट्रैफ़िक की तीव्रता (इंटरनेट राउटर से संस्थान राउटर तक) है

(15 अनुरोध/सेकंड)⯑(1 एमबिट्स/अनुरोध)/(15 एमबीपीएस)=1

LAN पर 0.15 की ट्रैफिक तीव्रता के परिणामस्वरूप आमतौर पर, अधिकतम दसियों मिलीसेकंड की देरी होती है; इसलिए, हम लैन विलंब की उपेक्षा कर सकते हैं। हालाँकि, जैसा कि खंड 1.4.2 में चर्चा की गई है, जैसे-जैसे ट्रैफ़िक की तीव्रता 1 के करीब पहुँचती है (जैसा कि चित्र 2.12 में एक्सेस लिंक के मामले में है), लिंक पर विलंब बहुत बड़ा हो जाता है और बिना सीमा के बढ़ता है। इस प्रकार, अनुरोधों को संतुष्ट करने के लिए औसत प्रतिक्रिया समय मिनटों के क्रम में होने वाला है, यदि अधिक नहीं, जो संस्थान के उपयोगकर्ताओं के लिए अस्वीकार्य है। स्पष्ट रूप से कुछ किया जाना चाहिए।

एक संभावित समाधान यह है कि एक्सेस दर को 15 एमबीपीएस से बढ़ाकर 100 एमबीपीएस कर दिया जाए। इससे एक्सेस लिंक पर ट्रैफ़िक की तीव्रता 0.15 तक कम हो जाएगी, जिसका मतलब है कि दो राउटर के बीच देरी नगण्य होगी।
इस मामले में, कुल प्रतिक्रिया समय लगभग दो सेकंड होगा, यानी इंटरनेट देरी। लेकिन इस समाधान का यह भी मतलब है कि संस्थान को अपने एक्सेस लिंक को 15 एमबीपीएस से 100 एमबीपीएस तक अपग्रेड करना होगा, जो एक महंगा प्रस्ताव है।

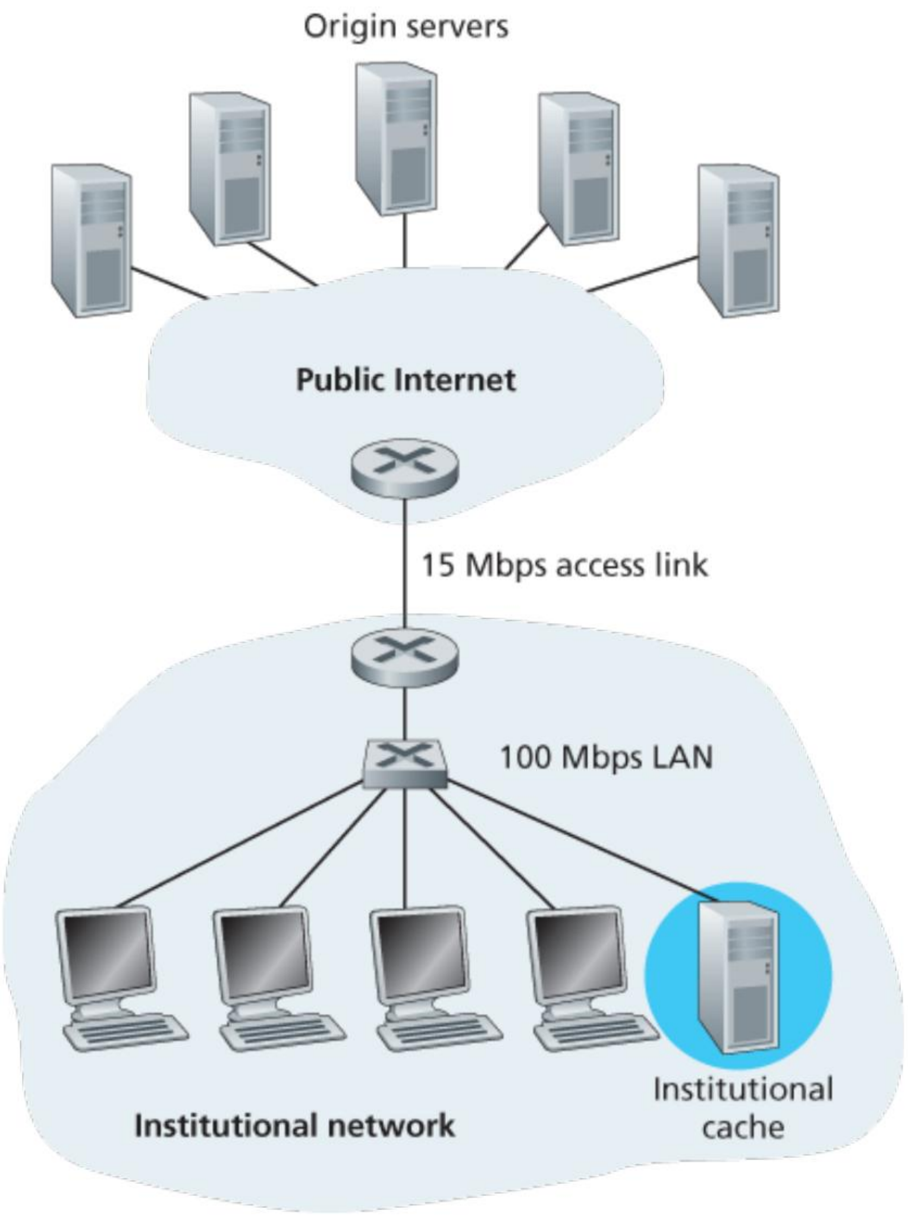
अब एक्सेस लिंक को अपग्रेड न करने बल्कि इसके स्थान पर वेब कैश स्थापित करने के वैकल्पिक समाधान पर विचार करें संस्थागत नेटवर्क में। इस समाधान को चित्र 2.13 में दर्शाया गया है। हिट दरें - कैश द्वारा संतुष्ट किए जाने वाले अनुरोधों का अंश - आमतौर पर व्यवहार में 0.2 से 0.7 तक होती हैं। उदाहरण के लिए, मान लें कि कैश इस संस्थान के लिए 0.4 की हिट दर प्रदान करता है। क्योंकि क्लाइंट और कैश एक ही हाई-स्पीड LAN से जुड़े हुए हैं, इसलिए कैश द्वारा 40 प्रतिशत अनुरोध लगभग तुरंत, यानी 10 मिलीसेकंड के भीतर संतुष्ट किए जाएँगे। फिर भी, शेष 60 प्रतिशत अनुरोधों को अभी भी मूल सर्वर द्वारा संतुष्ट किए जाने की आवश्यकता है। लेकिन एक्सेस लिंक से केवल 60 प्रतिशत अनुरोधित ऑब्जेक्ट के गुजरने पर, एक्सेस लिंक पर ट्रैफ़िक की तीव्रता 1.0 से 0.6 तक कम हो जाती है।

आम तौर पर, 0.8 से कम ट्रैफ़िक तीव्रता 15 एमबीपीएस लिंक पर एक छोटी देरी, मान लीजिए, दसियों मिलीसेकंड के बराबर होती है। यह देरी दो सेकंड की इंटरनेट देरी की तुलना में नगण्य है। इन विचारों को देखते हुए, औसत देरी इसलिए है

0.4⯑(0.01 सेकंड)+0.6⯑(2.01 सेकंड)

जो 1.2 सेकंड से थोड़ा ज़्यादा है। इस प्रकार, यह दूसरा समाधान पहले समाधान की तुलना में और भी कम प्रतिक्रिया समय प्रदान करता है, और इसके लिए संस्थान की आवश्यकता नहीं होती है

Machine Translated by Google



चित्र 2.13 संस्थागत नेटवर्क में कैश जोड़ना

इंटरनेट से अपने लिंक को अपग्रेड करने के लिए। संस्था को, निश्चित रूप से, एक वेब कैश खरीदना और इंस्टॉल करना होगा। लेकिन यह लागत कम है - कई कैश सार्वजनिक-डोमेन सॉफ़्टवेयर का उपयोग करते हैं जो सस्ते पीसी पर चलते हैं।

कंटेंट डिस्ट्रीब्यूशन नेटवर्क (CDN) के उपयोग के माध्यम से , वेब कैश इंटरनेट में तेजी से महत्वपूर्ण भूमिका निभा रहे हैं। एक CDN कंपनी पूरे इंटरनेट में कई भौगोलिक रूप से वितरित कैश स्थापित करती है, जिससे अधिकांश ट्रैफ़िक स्थानीयकृत हो जाता है। साझा CDN (जैसे Akamai और Limelight) और समर्पित CDN (जैसे Google और Netflix) हैं। हम CDN के बारे में अधिक विस्तार से चर्चा करेंगे

अनुभाग 2.6 में ।

सशर्त GET

हालाँकि कैशिंग उपयोगकर्ता द्वारा अनुभव किए जाने वाले प्रतिक्रिया समय को कम कर सकता है, लेकिन यह एक नई समस्या पेश करता है - कैश में रहने वाले ऑब्जेक्ट की कॉपी पुरानी हो सकती है। दूसरे शब्दों में, वेब सर्वर में रखे गए ऑब्जेक्ट को क्लाइंट पर कैश किए जाने के बाद से संशोधित किया जा सकता है। सौभाग्य से, HTTP में एक तंत्र है जो कैश को यह सत्यापित करने की अनुमति देता है कि उसके ऑब्जेक्ट अद्यतित हैं। इस तंत्र को सशर्त GET कहा जाता है।

एक HTTP अनुरोध संदेश एक तथाकथित सशर्त GET संदेश है यदि (1) अनुरोध संदेश का उपयोग करता है

GET विधि और (2) अनुरोध संदेश में If-Modified-Since: हेडर लाइन शामिल है।

यह समझाने के लिए कि सशर्त GET कैसे काम करता है, आइए एक उदाहरण देखें। सबसे पहले, अनुरोध करने वाले ब्राउज़र की ओर से, प्रॉक्सी कैश वेब सर्वर को एक अनुरोध संदेश भेजता है:

```
/fruit/kiwi.gif HTTP/1.1 प्राप्त करें
होस्ट: www.exotiquecuisine.com
```

दूसरा, वेब सर्वर अनुरोधित ऑब्जेक्ट के साथ कैश को एक प्रतिक्रिया संदेश भेजता है:

```
HTTP/1.1 200 ठीक
दिनांक: शनिवार, 3 अक्टूबर 2015 15:39:29
सर्वर: अपाचे/1.3.0 (यूनिक्स)
अंतिम संशोधन: बुध, 9 सितम्बर 2015 09:23:24
सामग्री-प्रकार: छवि/gif

(डेटा डेटा डेटा डेटा डेटा ...)
```

कैश ऑब्जेक्ट को अनुरोध करने वाले ब्राउज़र को अग्रेषित करता है, लेकिन ऑब्जेक्ट को स्थानीय रूप से कैश भी करता है। महत्वपूर्ण बात यह है कि कैश ऑब्जेक्ट के साथ अंतिम संशोधित तिथि भी संग्रहीत करता है। तीसरा, एक सप्ताह बाद, दूसरा ब्राउज़र कैश के माध्यम से उसी ऑब्जेक्ट का अनुरोध करता है, और ऑब्जेक्ट अभी भी कैश में है। चूँकि इस ऑब्जेक्ट को पिछले सप्ताह वेब सर्वर पर संशोधित किया गया हो सकता है, इसलिए कैश एक सशर्त GET जारी करके अप-टू-डेट जाँच करता है। विशेष रूप से, कैश भेजता है:

```
/fruit/kiwi.gif HTTP/1.1 प्राप्त करें
होस्ट: www.exotiquecuisine.com
यदि-संशोधित-से: बुध, 9 सितम्बर 2015 09:23:24
```

ध्यान दें कि If-modified-since: हेडर लाइन का मान, के मान के बिल्कुल बराबर है।

अंतिम-संशोधित: हेडर लाइन जो सर्वर द्वारा एक सप्ताह पहले भेजी गई थी। यह सशर्त GET सर्वर को ऑब्जेक्ट को केवल तभी भेजने के लिए कह रहा है जब ऑब्जेक्ट को निर्दिष्ट तिथि के बाद से संशोधित किया गया हो।

मान लीजिए कि 9 Sep 2015 09:23:24 के बाद से ऑब्जेक्ट को संशोधित नहीं किया गया है। फिर, चौथा, वेब सर्वर कैश को एक प्रतिक्रिया संदेश भेजता है:

```
HTTP/1.1 304 संशोधित नहीं
दिनांक: शनिवार, 10 अक्टूबर 2015 15:39:29
सर्वर: अपाचे/1.3.0 (यूनिक्स)

(खाली इकाई निकाय)
```

हम देखते हैं कि सशर्त GET के जवाब में, वेब सर्वर अभी भी एक प्रतिक्रिया संदेश भेजता है लेकिन प्रतिक्रिया संदेश में अनुरोधित ऑब्जेक्ट को शामिल नहीं करता है। अनुरोधित ऑब्जेक्ट को शामिल करने से केवल बैंडविड्थ बर्बाद होगी और उपयोगकर्ता द्वारा अनुभव किया जाने वाला प्रतिक्रिया समय बढ़ेगा, खासकर अगर ऑब्जेक्ट बड़ा हो। नोट

इस अंतिम प्रतिक्रिया संदेश की स्थिति पंक्ति में 304 Not Modified लिखा **है, जो कैश को बताता है कि वह आगे बढ़ सकता है और ऑब्जेक्ट की अपनी (प्रॉक्सी कैश की) कैश की गई प्रतिलिपि को अनुरोधकर्ता ब्राउज़र को अग्रेषित कर सकता है।**

यह HTTP, पहला इंटरनेट प्रोटोकॉल (एप्लिकेशन-लेयर प्रोटोकॉल) के बारे में हमारी चर्चा को समाप्त करता है, जिसका हमने विस्तार से अध्ययन किया है। हमने HTTP संदेशों के प्रारूप और इन संदेशों को भेजे और प्राप्त किए जाने के दौरान वेब क्लाइंट और सर्वर द्वारा की जाने वाली कार्रवाइयों को देखा है। हमने वेब के एप्लिकेशन इंफ्रास्ट्रक्चर का भी थोड़ा अध्ययन किया है, जिसमें कैश, कुकीज़ और बैक-एंड डेटाबेस शामिल हैं, जो सभी किसी न किसी तरह से जुड़े हुए हैं

HTTP प्रोटोकॉल.

2.3 इंटरनेट पर इलेक्ट्रॉनिक मेल

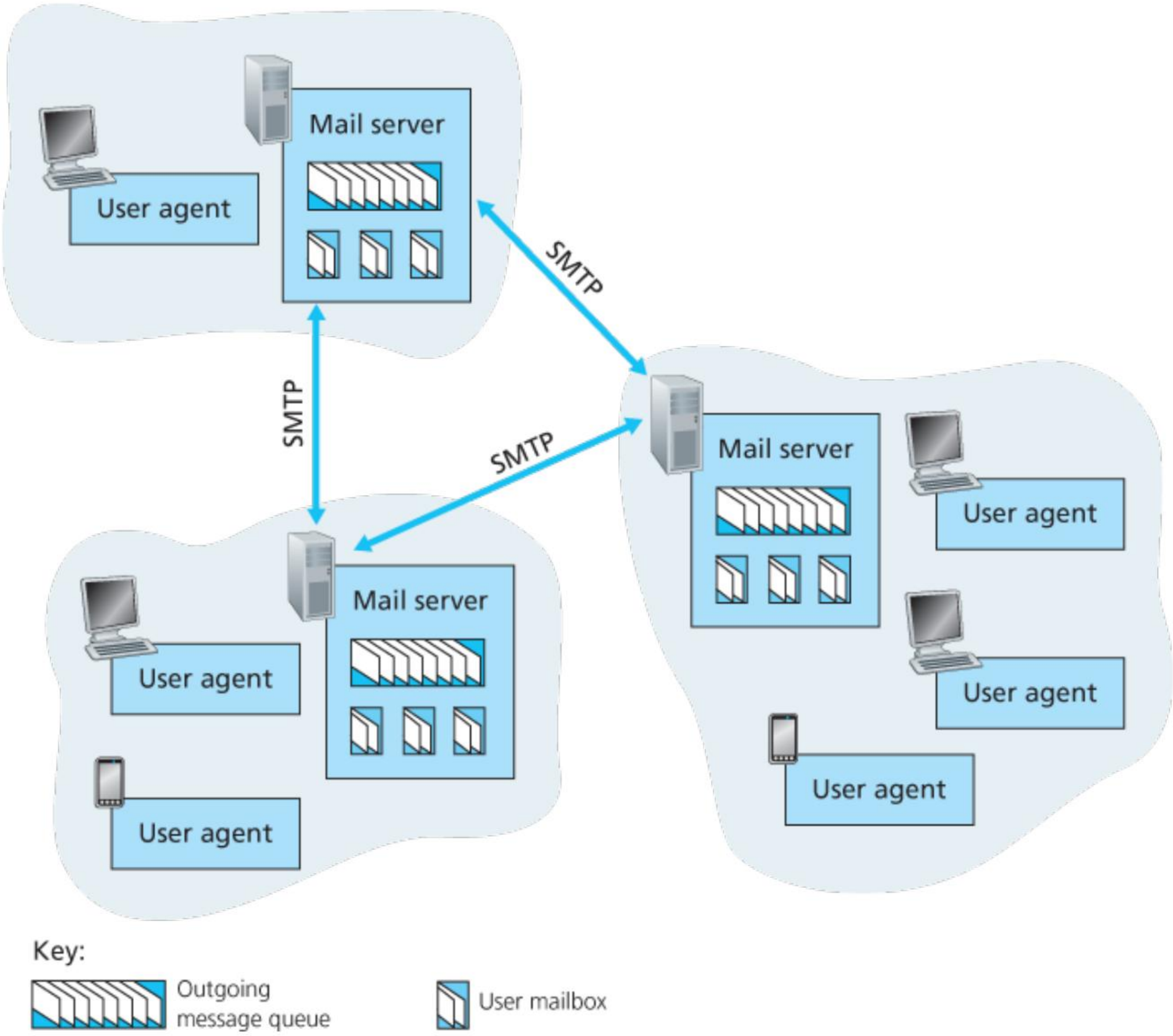
इलेक्ट्रॉनिक मेल इंटरनेट की शुरुआत से ही मौजूद है। यह सबसे लोकप्रिय एप्लीकेशन था
जब इंटरनेट अपनी प्रारंभिक अवस्था में था [सेगलर 1998], और पिछले कुछ वर्षों में यह अधिक विस्तृत और शक्तिशाली हो गया है। यह इंटरनेट के सबसे महत्वपूर्ण और उपयोग किए जाने वाले अनुप्रयोगों में से एक है।

सामान्य डाक मेल की तरह, ई-मेल एक अतुल्यकालिक संचार माध्यम है - लोग जब चाहें संदेश भेजते और पढ़ते हैं, बिना दूसरे लोगों के शेड्यूल से तालमेल बिठाए। डाक मेल के विपरीत, इलेक्ट्रॉनिक मेल तेज़, वितरित करने में आसान और सस्ता है। आधुनिक ई-मेल में कई शक्तिशाली विशेषताएं हैं, जिनमें अटैचमेंट वाले संदेश, हाइपरलिंक, HTML-स्वरूपित टेक्स्ट और एम्बेडेड फ़ोटो शामिल हैं।

इस खंड में, हम उन एप्लिकेशन-लेयर प्रोटोकॉल की जांच करेंगे जो इंटरनेट ई-मेल के मूल में हैं। लेकिन इन प्रोटोकॉल की गहन चर्चा में जाने से पहले, आइए इंटरनेट मेल सिस्टम और इसके प्रमुख घटकों का उच्च-स्तरीय दृश्य देखें।

चित्र 2.14 इंटरनेट मेल सिस्टम का उच्च-स्तरीय दृश्य प्रस्तुत करता है। इस आरेख से हम देखते हैं कि इसके तीन प्रमुख घटक हैं: उपयोगकर्ता एजेंट, मेल सर्वर और सरल मेल ट्रांसफ़र प्रोटोकॉल (SMTP)।
अब हम इनमें से प्रत्येक घटक का वर्णन एक प्रेषक, एलिस द्वारा प्राप्तकर्ता, बॉब को ई-मेल संदेश भेजने के संदर्भ में करते हैं। उपयोगकर्ता एजेंट उपयोगकर्ताओं को संदेशों को पढ़ने, उत्तर देने, अग्रेषित करने, सहेजने और लिखने की अनुमति देते हैं। Microsoft Outlook और Apple Mail ई-मेल के लिए उपयोगकर्ता एजेंटों के उदाहरण हैं। जब एलिस अपना संदेश लिखना समाप्त कर लेती है, तो उसका उपयोगकर्ता एजेंट संदेश को उसके मेल सर्वर पर भेजता है, जहाँ संदेश को मेल सर्वर की आउटगोइंग संदेश कतार में रखा जाता है। जब बॉब कोई संदेश पढ़ना चाहता है, तो उसका उपयोगकर्ता एजेंट उसके मेल सर्वर में उसके मेलबॉक्स से संदेश प्राप्त करता है।

मेल सर्वर ई-मेल इंफ्रास्ट्रक्चर का मुख्य हिस्सा होते हैं। बॉब जैसे प्रत्येक प्राप्तकर्ता के पास मेल सर्वर में से किसी एक में स्थित एक मेलबॉक्स होता है। बॉब का मेलबॉक्स ईमेल सर्वर को प्रबंधित और प्रबंधित करता है।



चित्र 2.14 इंटरनेट ई-मेल प्रणाली का उच्च-स्तरीय दृश्य

उसे भेजे गए संदेशों को बनाए रखता है। एक सामान्य संदेश प्रेषक के उपयोगकर्ता एजेंट में अपनी यात्रा शुरू करता है, प्रेषक के मेल सर्वर तक जाता है, और प्राप्तकर्ता के मेल सर्वर तक जाता है, जहाँ इसे प्राप्तकर्ता के मेलबॉक्स में जमा किया जाता है। जब बॉब अपने मेलबॉक्स में संदेशों तक पहुँचना चाहता है, तो उसका मेलबॉक्स वाला मेल सर्वर बॉब को प्रमाणित करता है (उपयोगकर्ता नाम और पासवर्ड के साथ)। ऐलिस का मेल सर्वर

बॉब के मेल सर्वर में विफलताओं से भी निपटना होगा। अगर ऐलिस का सर्वर बॉब के सर्वर तक मेल नहीं पहुंचा पाता है,
ऐलिस का सर्वर संदेश को संदेश कतार **में रखता है** तथा बाद में संदेश को स्थानांतरित करने का प्रयास करता है।
पुनः प्रयास प्रायः हर 30 मिनट में किया जाता है; यदि कई दिनों के बाद भी सफलता नहीं मिलती, तो सर्वर संदेश को हटा देता है और प्रेषक (ऐलिस) को ई-मेल संदेश भेजकर सूचित करता है।

SMTP इंटरनेट इलेक्ट्रॉनिक मेल के लिए मुख्य एप्लीकेशन-लेयर प्रोटोकॉल है। यह प्रेषक के मेल सर्वर से प्राप्तकर्ता के मेल सर्वर तक मेल स्थानांतरित करने के लिए TCP की विश्वसनीय डेटा ट्रांसफर सेवा का उपयोग करता है। अधिकांश एप्लीकेशन-लेयर प्रोटोकॉल की तरह, SMTP के भी दो पक्ष हैं: एक क्लाइंट साइड, जो प्रेषक के मेल सर्वर पर निष्पादित होता है, और एक सर्वर साइड, जो प्राप्तकर्ता के मेल सर्वर पर निष्पादित होता है। SMTP के क्लाइंट और सर्वर दोनों पक्ष प्रत्येक मेल सर्वर पर चलते हैं। जब कोई मेल सर्वर अन्य मेल सर्वर को मेल भेजता है, तो यह एक सर्वर के रूप में कार्य करता है।

एक SMTP क्लाइंट। जब एक मेल सर्वर अन्य मेल सर्वर से मेल प्राप्त करता है, तो यह एक SMTP सर्वर के रूप में कार्य करता है।

2.3.1 एसएमटीपी

RFC 5321 में परिभाषित SMTP, इंटरनेट इलेक्ट्रॉनिक मेल का मूल है। जैसा कि ऊपर बताया गया है, SMTP प्रेषक के मेल सर्वर से प्राप्तकर्ताओं के मेल सर्वर पर संदेश स्थानांतरित करता है। SMTP HTTP से बहुत पुराना है। (मूल SMTP RFC 1982 का है, और SMTP उससे बहुत पहले से मौजूद था।) हालाँकि SMTP में कई अद्भुत गुण हैं, जैसा कि इंटरनेट में इसकी सर्वव्यापकता से प्रमाणित होता है, फिर भी यह एक विरासत तकनीक है जिसमें कुछ पुरातन विशेषताएँ हैं। उदाहरण के लिए, यह सभी मेल संदेशों के मुख्य भाग (सिर्फ़ हेडर नहीं) को सरल 7-बिट ASCII तक सीमित करता है। यह प्रतिबंध 1980 के दशक की शुरुआत में समझ में आता था जब ट्रांसमिशन क्षमता कम थी और कोई भी बड़ी अनुलग्नक या बड़ी छवि, ऑडियो या वीडियो फ़ाइलें ई-मेल नहीं कर रहा था। लेकिन आज, मल्टीमीडिया युग में, 7-बिट ASCII प्रतिबंध थोड़ा कष्टदायक है - इसके लिए SMTP पर भेजे जाने से पहले बाइनरी मल्टीमीडिया डेटा को ASCII में एनकोड करना आवश्यक है; और इसके लिए SMTP ट्रांसपोर्ट के बाद संबंधित ASCII संदेश को वापस बाइनरी में डीकोड करना आवश्यक है। याद रखें

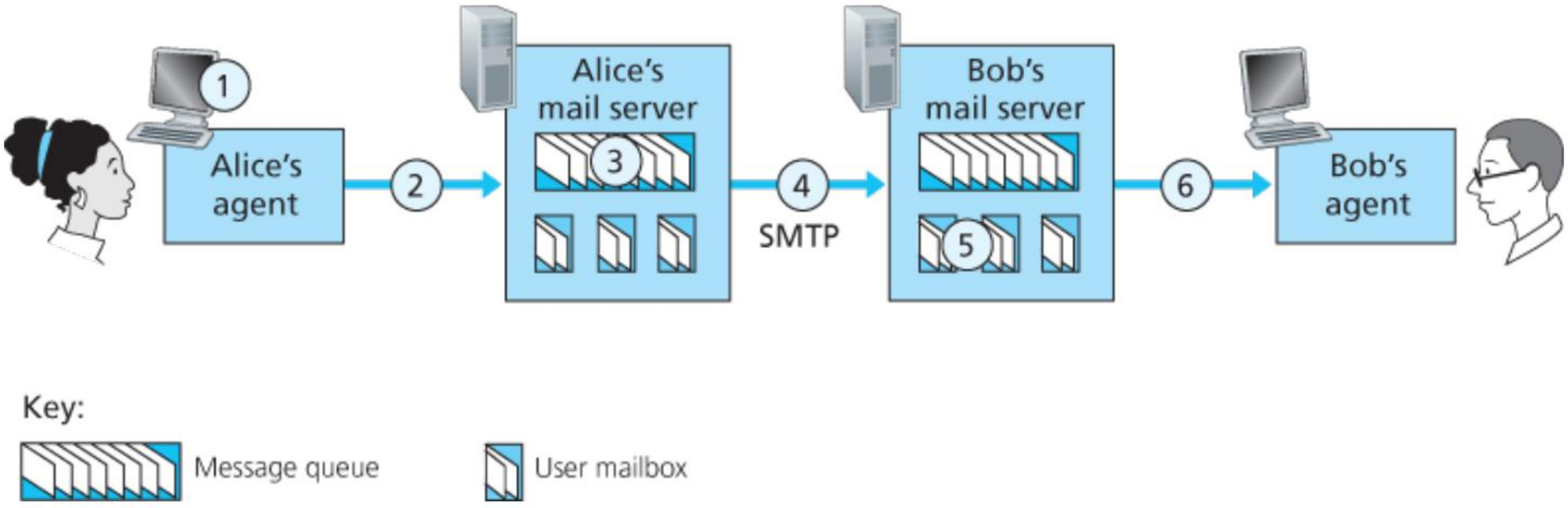
धारा 2.2 से स्पष्ट है कि HTTP को स्थानांतरण से पहले मल्टीमीडिया डेटा को ASCII एनकोड करने की आवश्यकता नहीं होती है।

SMTP के बुनियादी संचालन को समझाने के लिए, आइए एक सामान्य परिदृश्य पर चलते हैं। मान लीजिए कि ऐलिस बॉब को एक सरल ASCII संदेश भेजना चाहती है।

1. ऐलिस ई-मेल के लिए अपने उपयोगकर्ता एजेंट को आमंत्रित करती है, बॉब का ई-मेल पता प्रदान करती है (उदाहरण के लिए, बॉब@someschool.edu), एक संदेश बनाता है, और उपयोगकर्ता एजेंट को संदेश भेजने का निर्देश देता है संदेश।
2. ऐलिस का उपयोगकर्ता एजेंट संदेश को उसके मेल सर्वर पर भेजता है, जहां इसे संदेश कतार में रखा जाता है।
3. एलिस के मेल सर्वर पर चलने वाला SMTP का क्लाइंट साइड, मैसेज कतार में संदेश देखता है। यह बॉब के मेल सर्वर पर चलने वाले SMTP सर्वर से TCP कनेक्शन खोलता है।
4. कुछ प्रारंभिक SMTP हैंडशेकिंग के बाद, SMTP क्लाइंट एलिस के संदेश को TCP में भेजता है कनेक्शन.
5. बॉब के मेल सर्वर पर, SMTP का सर्वर साइड संदेश प्राप्त करता है। फिर बॉब का मेल सर्वर संदेश को बॉब के मेलबॉक्स में डालता है।
6. बॉब अपनी सुविधानुसार संदेश पढ़ने के लिए अपने उपयोगकर्ता एजेंट को आमंत्रित करता है।

परिदृश्य का सारांश चित्र 2.15 में दिया गया है।

यह ध्यान रखना महत्वपूर्ण है कि SMTP आम तौर पर मेल भेजने के लिए मध्यवर्ती मेल सर्वर का उपयोग नहीं करता है, भले ही दो मेल सर्वर दुनिया के विपरीत छोर पर स्थित हों। यदि ऐलिस का सर्वर हांगकांग में है और बॉब का सर्वर सेंट लुइस में है, तो TCP



चित्र 2.15 ऐलिस बॉब को संदेश भेजती है

कनेक्शन हांगकांग और सेंट लुइस सर्वर के बीच एक सीधा कनेक्शन है। विशेष रूप से, यदि बॉब का मेल सर्वर डाउन है, तो संदेश ऐलिस के मेल सर्वर में रहता है और एक नए प्रयास की प्रतीक्षा करता है - संदेश किसी मध्यवर्ती मेल सर्वर में नहीं रखा जाता है।

आइए अब इस बात पर करीब से नज़र डालें कि SMTP किसी संदेश को भेजने वाले मेल सर्वर से प्राप्त करने वाले मेल सर्वर पर कैसे स्थानांतरित करता है। हम देखेंगे कि SMTP प्रोटोकॉल में आमने-सामने मानवीय संपर्क के लिए उपयोग किए जाने वाले प्रोटोकॉल के साथ कई समानताएँ हैं। सबसे पहले, क्लाइंट SMTP (भेजने वाले मेल सर्वर होस्ट पर चल रहा है) में TCP द्वारा सर्वर SMTP (प्राप्त करने वाले मेल सर्वर होस्ट पर चल रहा है) पर पोर्ट 25 से कनेक्शन स्थापित किया जाता है। यदि सर्वर डाउन है, तो क्लाइंट बाद में फिर से प्रयास करता है। एक बार यह कनेक्शन स्थापित हो जाने के बाद, सर्वर और क्लाइंट कुछ एप्लिकेशन-लेयर हैंडशेकिंग करते हैं - जैसे कि मनुष्य अक्सर एक से दूसरे को जानकारी स्थानांतरित करने से पहले अपना परिचय देते हैं, SMTP क्लाइंट और सर्वर जानकारी स्थानांतरित करने से पहले अपना परिचय देते हैं। इस SMTP हैंडशेकिंग चरण के दौरान, SMTP क्लाइंट प्रेषक (संदेश बनाने वाला व्यक्ति) का ई-मेल पता और प्राप्तकर्ता का ई-मेल पता बताता है। एक बार जब SMTP क्लाइंट और सर्वर एक-दूसरे से अपना परिचय दे देते हैं, तो क्लाइंट संदेश भेजता है। SMTP बिना किसी त्रुटि के सर्वर तक संदेश पहुंचाने के लिए TCP की विश्वसनीय डेटा ट्रांसफर सेवा पर भरोसा कर सकता है। क्लाइंट फिर उसी TCP कनेक्शन पर इस प्रक्रिया को दोहराता है यदि उसके पास सर्वर को भेजने के लिए अन्य संदेश हैं; अन्यथा, यह TCP को कनेक्शन बंद करने का निर्देश देता है।

आइए अब एक SMTP क्लाइंट (C) के बीच आदान-प्रदान किए गए संदेशों की एक उदाहरण प्रतिलिपि पर नज़र डालें

और एक SMTP सर्वर (S)। क्लाइंट का होस्टनाम crepes.fr है और सर्वर का होस्टनाम है

hamburger.edu . C: से शुरू होने वाली ASCII टेक्स्ट लाइनें ठीक वही लाइनें हैं जो क्लाइंट अपने सर्वर पर भेजता है।

TCP सॉकेट, और S: से शुरू होने वाली ASCII टेक्स्ट लाइनें बिल्कुल वही लाइनें हैं जो सर्वर अपने TCP सॉकेट में भेजता है। TCP कनेक्शन स्थापित होते ही निम्नलिखित ट्रांसक्रिप्ट शुरू हो जाती है।

```
एस: 220 hamburger.edu
सी: हेलो crepes.fr
S: 250 नमस्ते crepes.fr, आपसे मिलकर खुशी हुई
```

C: मेल से: <alice@crepes.fr>

S: 250 alice@crepes.fr ... प्रेषक ठीक है

सी: आरसीपीटी टू: <bob@hamburger.edu>

S: 250 bob@hamburger.edu ... प्राप्तकर्ता ठीक है

सी: डेटा

एस: 354 एक पंक्ति में मेल दर्ज करें, जिसका अंत "." से हो

सी: क्या आपको केचप पसंद है?

सी: अचार के बारे में क्या ख्याल है?

सी: ।

S: 250 संदेश डिलीवरी के लिए स्वीकार किया गया

सी: छोड़ो

S: 221 hamburger.edu कनेक्शन बंद कर रहा है

उपरोक्त उदाहरण में, ग्राहक एक संदेश भेजता है (" क्या आपको केचप पसंद है? अचार? ") मेल सर्वर crepes.fr से मेल सर्वर hamburger.edu पर । संवाद के एक भाग के रूप में, ग्राहक ने पांच आदेश जारी किए: HELO (HELLO का संक्षिप्त रूप), MAIL FROM , आरसीपीटी टू , डेटा , और QUIT । ये आदेश स्वतः स्पष्ट हैं। क्लाइंट एक पंक्ति भी भेजता है जिसमें एक एकल पंक्ति होती है अवधि, जो सर्वर को भेजे गए संदेश के अंत को इंगित करती है। (ASCII शब्दावली में, प्रत्येक संदेश के अंत में सीआरएलएफ.सीआरएलएफ, जहां सीआर और एलएफ क्रमशः कैरिज रिटर्न और लाइन फीड के लिए हैं।) सर्वर प्रत्येक आदेश के लिए उत्तर जारी करता है, जिसमें प्रत्येक उत्तर में एक उत्तर कोड और कुछ (वैकल्पिक) अंग्रेजी-भाषा स्पष्टीकरण होता है। हम यहाँ उल्लेख करते हैं कि SMTP स्थायी कनेक्शन का उपयोग करता है: यदि भेजने वाला मेल यदि सर्वर के पास एक ही प्राप्तकर्ता मेल सर्वर को भेजने के लिए कई संदेश हैं, तो वह सभी संदेशों को भेज सकता है उसी TCP कनेक्शन पर। प्रत्येक संदेश के लिए, क्लाइंट एक नए MAIL के साथ प्रक्रिया शुरू करता है स्रोत: crepes.fr , एक पृथक अवधि के साथ संदेश के अंत को निर्दिष्ट करता है, और केवल QUIT जारी करता है सभी संदेश भेजे जाने के बाद.

यह अत्यधिक अनुशंसित है कि आप SMTP सर्वर के साथ सीधा संवाद करने के लिए टेलनेट का उपयोग करें। यह मुद्दा

टेलनेट सर्वरनाम 25

जहाँ serverName एक स्थानीय मेल सर्वर का नाम है। जब आप ऐसा करते हैं, तो आप बस एक सर्वर स्थापित कर रहे होते हैं। आपके स्थानीय होस्ट और मेल सर्वर के बीच TCP कनेक्शन। यह लाइन टाइप करने के बाद, आपको सर्वर से तुरंत 220 उत्तर प्राप्त करें। फिर SMTP कमांड HELO जारी करें , मेल से , आरसीपीटी टू , डेटा , सीआरएलएफ.सीआरएलएफ, और उचित समय पर छोड़ दें । यह भी अत्यधिक है इस अध्याय के अंत में प्रोग्रामिंग असाइनमेंट 3 करने की सलाह दी गई है। उस असाइनमेंट में, आप एक सरल उपयोगकर्ता एजेंट बनाएंगे जो SMTP के क्लाइंट साइड को लागू करता है। यह आपको ई-मेल भेजने की अनुमति देगा।

किसी स्थानीय मेल सर्वर के माध्यम से किसी मनमाने प्राप्तकर्ता को मेल संदेश भेजना।

2.3.2 HTTP के साथ तुलना

आइए अब संक्षेप में SMTP की तुलना HTTP से करें। दोनों प्रोटोकॉल का उपयोग फ़ाइलों को एक होस्ट से दूसरे होस्ट में स्थानांतरित करने के लिए किया जाता है: HTTP फ़ाइलों (जिन्हें ऑब्जेक्ट भी कहा जाता है) को वेब सर्वर से वेब क्लाइंट (आमतौर पर ब्राउज़र) में स्थानांतरित करता है; SMTP फ़ाइलों (यानी, ई-मेल संदेश) को एक मेल सर्वर से दूसरे मेल सर्वर में स्थानांतरित करता है।
फ़ाइलों को स्थानांतरित करते समय, लगातार HTTP और SMTP दोनों लगातार कनेक्शन का उपयोग करते हैं। इस प्रकार, दोनों प्रोटोकॉल में सामान्य विशेषताएँ हैं। हालाँकि, कुछ महत्वपूर्ण अंतर हैं। सबसे पहले, HTTP मुख्य रूप से एक पुल प्रोटोकॉल है - कोई व्यक्ति वेब सर्वर पर जानकारी लोड करता है और उपयोगकर्ता अपनी सुविधानुसार सर्वर से जानकारी खींचने के लिए HTTP का उपयोग करते हैं। विशेष रूप से, TCP कनेक्शन उस मशीन द्वारा शुरू किया जाता है जो फ़ाइल प्राप्त करना चाहती है। दूसरी ओर, SMTP मुख्य रूप से एक पुश प्रोटोकॉल है - भेजने वाला मेल सर्वर फ़ाइल को प्राप्त करने वाले मेल सर्वर पर धकेलता है। विशेष रूप से, TCP कनेक्शन उस मशीन द्वारा शुरू किया जाता है जो फ़ाइल भेजना चाहती है।

दूसरा अंतर, जिसका हमने पहले उल्लेख किया था, वह यह है कि SMTP के लिए प्रत्येक संदेश, जिसमें प्रत्येक संदेश का मुख्य भाग भी शामिल है, 7-बिट ASCII प्रारूप में होना आवश्यक है। यदि संदेश में ऐसे वर्ण हैं जो 7-बिट ASCII नहीं हैं (उदाहरण के लिए, उच्चारण के साथ फ्रेंच वर्ण) या बाइनरी डेटा (जैसे कि एक छवि फ़ाइल) शामिल है, तो संदेश को 7-बिट ASCII में एनकोड किया जाना चाहिए। HTTP डेटा इस प्रतिबंध को लागू नहीं करता है।

तीसरा महत्वपूर्ण अंतर यह है कि पाठ और छवियों (संभवतः छवियों के साथ) से युक्त दस्तावेज़ को कैसे प्रस्तुत किया जाता है।
अन्य मीडिया प्रकारों) को संभाला जाता है। जैसा कि हमने अनुभाग 2.2 में सीखा, HTTP प्रत्येक ऑब्जेक्ट को अपने स्वयं के HTTP प्रतिक्रिया संदेश में समाहित करता है। SMTP संदेश के सभी ऑब्जेक्ट को एक संदेश में रखता है।

2.3.3 मेल संदेश प्रारूप

जब ऐलिस बॉब को एक साधारण डाक पत्र लिखती है, तो वह पत्र के शीर्ष पर सभी प्रकार की परिधीय हेडर जानकारी शामिल कर सकती है, जैसे बॉब का पता, उसका अपना वापसी पता, और तारीख।
इसी तरह, जब एक व्यक्ति से दूसरे व्यक्ति को ई-मेल संदेश भेजा जाता है, तो परिधीय जानकारी वाला हेडर संदेश के मुख्य भाग से पहले आता है। यह परिधीय जानकारी हेडर लाइनों की एक श्रृंखला में समाहित होती है, जिसे RFC 5322 में परिभाषित किया गया है। हेडर लाइनें और संदेश का मुख्य भाग एक दूसरे से जुड़े होते हैं।
एक खाली लाइन (यानी, CRLF **द्वारा**) द्वारा अलग किया गया। RFC 5322 मेल हेडर लाइनों के लिए सटीक प्रारूप के साथ-साथ उनकी अर्थपूर्ण व्याख्याओं को निर्दिष्ट करता है। HTTP की तरह, प्रत्येक हेडर लाइन में पठनीय पाठ होता है, जिसमें एक कीवर्ड के बाद एक कोलन और उसके बाद एक मान होता है।
कुछ कीवर्ड आवश्यक और आवश्यक हैं

अन्य वैकल्पिक हैं। प्रत्येक हेडर में एक From: हेडर लाइन और एक To: हेडर लाइन होनी चाहिए; एक हेडर

इसमें Subject: हेडर लाइन के साथ-साथ अन्य वैकल्पिक हेडर लाइनें भी शामिल हो सकती हैं **। यह ध्यान रखना महत्वपूर्ण है कि**

ये हेडर लाइनें SMTP कमांड से अलग **हैं जिनका हमने** सेक्शन 2.4.1 **में अध्ययन किया था** (भले ही

उनमें कुछ सामान्य शब्द होते हैं जैसे "से" और "से")। उस अनुभाग में दिए गए आदेश SMTP हैंडशेकिंग प्रोटोकॉल का हिस्सा थे; इस अनुभाग में जाँची गई हेडर लाइनें मेल संदेश का हिस्सा हैं
स्वयं.

एक सामान्य संदेश शीर्षलेख इस प्रकार दिखता है:

```
प्रेषक: alice@crepes.fr
को: bob@hamburger.edu
विषय: जीवन के अर्थ की खोज।
```

संदेश हेडर के बाद एक खाली लाइन आती है; फिर संदेश बॉडी (ASCII में) आती है। आपको मेल सर्वर पर संदेश भेजने के लिए टेलनेट का उपयोग करना चाहिए जिसमें कुछ हेडर लाइनें शामिल हों, जिनमें शामिल हैं
विषय: हेडर लाइन। ऐसा करने के लिए, टेलनेट सर्वरनाम 25 जारी करें, जैसा कि अनुभाग 2.4.1 में चर्चा की गई है ।

2.3.4 मेल एक्सेस प्रोटोकॉल

एक बार जब SMTP एलिस के मेल सर्वर से बॉब के मेल सर्वर तक संदेश पहुंचाता है, तो संदेश बॉब के मेलबॉक्स में डाल दिया जाता है। इस चर्चा के दौरान हमने मौन रूप से यह मान लिया है कि बॉब सर्वर होस्ट पर लॉग इन करके और फिर उस होस्ट पर चलने वाले मेल रीडर को निष्पादित करके अपना मेल पढ़ता है। 1990 के दशक की शुरुआत तक यह काम करने का मानक तरीका था। लेकिन आज, मेल एक्सेस क्लाइंट-सर्वर आर्किटेक्चर का उपयोग करता है - सामान्य उपयोगकर्ता क्लाइंट के साथ ई-मेल पढ़ता है जो उपयोगकर्ता के अंतिम सिस्टम पर निष्पादित होता है, उदाहरण के लिए, एक कार्यालय पीसी, एक लैपटॉप या एक स्मार्टफोन पर। स्थानीय पीसी पर मेल क्लाइंट निष्पादित करके, उपयोगकर्ता मल्टीमीडिया संदेशों और अनुलग्नकों को देखने की क्षमता सहित सुविधाओं के एक समृद्ध सेट का आनंद लेते हैं।

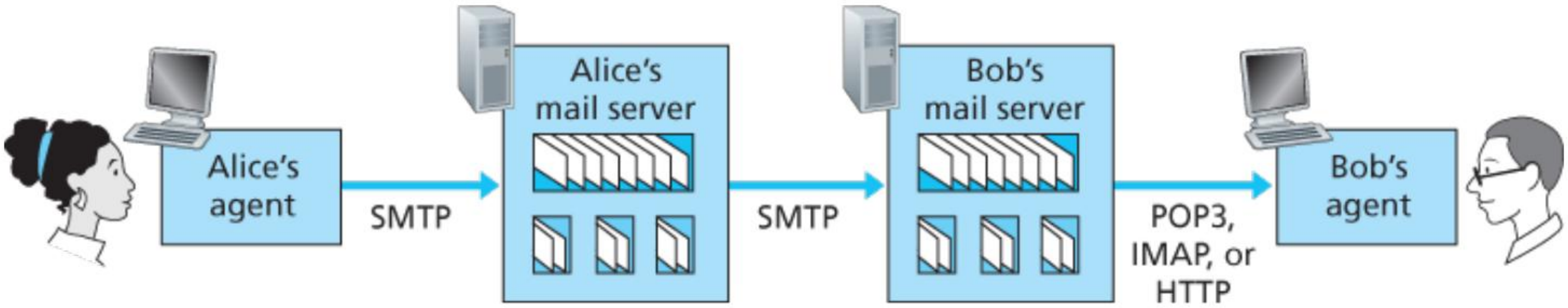
यह देखते हुए कि बॉब (प्राप्तकर्ता) अपने स्थानीय पीसी पर अपने उपयोगकर्ता एजेंट को निष्पादित करता है, उसके स्थानीय पीसी पर भी मेल सर्वर रखने पर विचार करना स्वाभाविक है। इस दृष्टिकोण के साथ, ऐलिस का मेल सर्वर बॉब के पीसी से सीधे संवाद करेगा। हालाँकि, इस दृष्टिकोण के साथ एक समस्या है। याद रखें कि एक मेल सर्वर प्रबंधित करता है

मेलबॉक्स और SMTP के क्लाइंट और सर्वर साइड चलाता है। अगर बॉब का मेल सर्वर उसके लोकल पर रहता
पीसी, तो बॉब के पीसी को हमेशा चालू रहना होगा, और इंटरनेट से जुड़ा रहना होगा, ताकि नया मेल प्राप्त हो सके, जो किसी भी समय आ सकता है। यह कई इंटरनेट उपयोगकर्ताओं के लिए अव्यावहारिक है। इसके बजाय, एक सामान्य उपयोगकर्ता स्थानीय पीसी पर एक उपयोगकर्ता एजेंट चलाता है, लेकिन हमेशा चालू रहने वाले साझा मेल सर्वर पर संग्रहीत अपने मेलबॉक्स तक पहुँचता है। यह मेल सर्वर अन्य उपयोगकर्ताओं के साथ साझा किया जाता है और आमतौर पर उपयोगकर्ता के ISP (उदाहरण के लिए, विश्वविद्यालय या कंपनी) द्वारा बनाए रखा जाता है।

अब आइए उस पथ पर विचार करें जो एक ई-मेल संदेश एलिस से बॉब को भेजे जाने पर अपनाता है। हमने अभी सीखा है कि पथ के किसी बिंदु पर ई-मेल संदेश को बॉब के मेल सर्वर में जमा किया जाना चाहिए। यह एलिस के उपयोगकर्ता एजेंट द्वारा संदेश को सीधे बॉब के मेल सर्वर पर भेजने से आसानी से किया जा सकता है। और

यह SMTP के साथ किया जा सकता है - वास्तव में, SMTP को एक होस्ट से दूसरे होस्ट तक ई-मेल भेजने के लिए डिज़ाइन किया गया है।
दूसरा। हालाँकि, आम तौर पर प्रेषक का उपयोगकर्ता एजेंट प्राप्तकर्ता के मेल से सीधे संवाद नहीं करता है

सर्वर। इसके बजाय, जैसा कि चित्र 2.16 में दिखाया गया है, ऐलिस का उपयोगकर्ता एजेंट ई-मेल संदेश को उसके मेल सर्वर में पुश करने के लिए SMTP का उपयोग करता है, फिर ऐलिस का मेल सर्वर ई-मेल संदेश को बॉब के मेल सर्वर पर रिले करने के लिए SMTP (SMTP क्लाइंट के रूप में) का उपयोग करता है। दो-चरणीय प्रक्रिया क्यों? मुख्य रूप से क्योंकि बिना रिले किए
ऐलिस के मेल सर्वर, ऐलिस के उपयोगकर्ता एजेंट के पास किसी अप्राप्य गंतव्य तक पहुंचने का कोई उपाय नहीं है



चित्र 2.16 ई-मेल प्रोटोकॉल और उनकी संचार इकाइयाँ

मेल सर्वर। एलिस द्वारा ई-मेल को पहले अपने मेल सर्वर में जमा करने से, एलिस का मेल सर्वर बार-बार बॉब के मेल सर्वर को संदेश भेजने का प्रयास कर सकता है, मान लीजिए हर 30 मिनट में, जब तक कि बॉब का मेल सर्वर चालू नहीं हो जाता। (और अगर एलिस का मेल सर्वर बंद है, तो उसके पास अपने सिस्टम एडमिनिस्ट्रेटर से शिकायत करने का विकल्प है!) SMTP RFC परिभाषित करता है कि SMTP कमांड का उपयोग कई SMTP सर्वरों पर संदेश रिले करने के लिए कैसे किया जा सकता है।

लेकिन पहेली में अभी भी एक टुकड़ा गायब है! बॉब जैसा प्राप्तकर्ता, अपने स्थानीय पीसी पर एक उपयोगकर्ता एजेंट चलाकर, अपने संदेशों को कैसे प्राप्त करता है, जो बॉब के आईएसपी के भीतर एक मेल सर्वर में बैठे हैं? ध्यान दें कि बॉब का उपयोगकर्ता एजेंट संदेशों को प्राप्त करने के लिए SMTP का उपयोग नहीं कर सकता क्योंकि संदेशों को प्राप्त करना एक पुल ऑपरेशन है, जबकि SMTP एक पुश प्रोटोकॉल है। पहेली एक विशेष मेल एक्सेस प्रोटोकॉल को पेश करके पूरी होती है जो बॉब के मेल सर्वर से उसके स्थानीय पीसी पर संदेश स्थानांतरित करता है। वर्तमान में कई लोकप्रिय मेल एक्सेस प्रोटोकॉल हैं, जिनमें पोस्ट ऑफिस प्रोटोकॉल-संस्करण 3 (POP3), इंटरनेट मेल एक्सेस प्रोटोकॉल (IMAP), और HTTP शामिल हैं।

चित्र 2.16 इंटरनेट मेल के लिए उपयोग किए जाने वाले प्रोटोकॉल का सारांश प्रदान करता है: SMTP का उपयोग प्रेषक के मेल सर्वर से प्राप्तकर्ता के मेल सर्वर पर मेल स्थानांतरित करने के लिए किया जाता है; SMTP का उपयोग प्रेषक के उपयोगकर्ता एजेंट से प्रेषक के मेल सर्वर पर मेल स्थानांतरित करने के लिए भी किया जाता है। मेल एक्सेस प्रोटोकॉल, जैसे कि POP3, का उपयोग प्राप्तकर्ता के मेल सर्वर से प्राप्तकर्ता के उपयोगकर्ता एजेंट पर मेल स्थानांतरित करने के लिए किया जाता है।

पॉप 3

POP3 एक अत्यंत सरल मेल एक्सेस प्रोटोकॉल है। इसे [RFC 1939] में परिभाषित किया गया है, जो संक्षिप्त और काफी पठनीय है। चूँकि प्रोटोकॉल बहुत सरल है, इसलिए इसकी कार्यक्षमता सीमित है। POP3 तब शुरू होता है जब उपयोगकर्ता एजेंट (क्लाइंट) पोर्ट 110 पर मेल सर्वर (सर्वर) से TCP कनेक्शन खोलता है।
TCP के साथ

कनेक्शन स्थापित होने के बाद, POP3 तीन चरणों से गुजरता है: प्राधिकरण, लेनदेन और अद्यतन।

पहले चरण, प्राधिकरण के दौरान, उपयोगकर्ता एजेंट उपयोगकर्ता नाम और पासवर्ड (स्पष्ट रूप में) भेजता है

उपयोगकर्ता को प्रमाणित करें। दूसरे चरण, लेनदेन के दौरान, उपयोगकर्ता एजेंट संदेशों को पुनः प्राप्त करता है; साथ ही

इस चरण के दौरान, उपयोगकर्ता एजेंट संदेशों को हटाने के लिए चिह्नित कर सकता है, हटाने के निशान हटा सकता है, और प्राप्त कर सकता है

मेल सांख्यिकी। तीसरा चरण, अद्यतन, क्लाइंट द्वारा क्विट कमांड जारी करने के बाद होता है, जिससे POP3 सत्र समाप्त हो जाता है; इस समय, मेल सर्वर उन संदेशों को हटा देता है जिन्हें हटाने के लिए चिह्नित किया गया था।

POP3 लेनदेन में, उपयोगकर्ता एजेंट आदेश जारी करता है, और सर्वर प्रत्येक आदेश का जवाब देता है

उत्तर के साथ। दो संभावित प्रतिक्रियाएँ हैं: +OK (कभी-कभी सर्वर-टू-क्लाइंट डेटा के बाद),

सर्वर द्वारा यह इंगित करने के लिए उपयोग किया जाता है कि पिछला कमांड ठीक था; तथा -ERR यह इंगित करता है कि पिछले कमांड में कुछ , सर्वर द्वारा उपयोग किया जाता है गड़बड़ थी।

प्राधिकरण चरण में दो मुख्य कमांड हैं: user <username> और pass <password> ।

इन दो आदेशों को स्पष्ट करने के लिए, हमारा सुझाव है कि आप पोर्ट का उपयोग करके सीधे POP3 सर्वर में टेलनेट करें

110 पर जाएँ और ये कमांड जारी करें। मान लीजिए कि mailServer आपके मेल सर्वर का नाम है। आपको कुछ ऐसा दिखाई देगा:

```
टेलनेट मेल सर्वर 110
+ठीक है POP3 सर्वर तैयार है
उपयोगकर्ता बॉब
+ठीक
भूखे पेट
+ठीक उपयोगकर्ता सफलतापूर्वक लॉग ऑन हुआ
```

यदि आप किसी आदेश की वर्तनी गलत लिखते हैं, तो POP3 सर्वर -ERR संदेश के साथ उत्तर देगा।

अब आइए लेनदेन चरण पर एक नज़र डालें। POP3 का उपयोग करने वाले उपयोगकर्ता एजेंट को अक्सर (उपयोगकर्ता द्वारा) "डाउनलोड और डिलीट" या "डाउनलोड और रखें" के लिए कॉन्फ़िगर किया जा सकता है। POP3 उपयोगकर्ता एजेंट द्वारा जारी किए गए आदेशों का क्रम इस बात पर निर्भर करता है कि उपयोगकर्ता एजेंट इन दो में से किस मोड में काम कर रहा है। डाउनलोड-

और-डिलीट मोड में, उपयोगकर्ता एजेंट सूची **जारी करेगा** , रिट्र , और डिलीट कमांड। उदाहरण के तौर पर,

मान लीजिए कि उपयोगकर्ता के मेलबॉक्स में दो संदेश हैं। नीचे दिए गए संवाद में, C: (जिसका अर्थ है

क्लाइंट) यूजर एजेंट है और S: (सर्वर के लिए खड़ा है) मेल सर्वर है। लेनदेन दिखेगा

कुछ इस तरह:

```
सी: सूची
एस: 1 498
एस: 2 912
```

एस: ।

सी: रिट्र 1

एस: (ब्ला ब्ला ...)

एस: ....

एस: ....ब्ला)

एस: ।

सी: डेले 1

सी: रिट्र 2

एस: (ब्ला ब्ला ...)

एस: ....

एस: ....ब्ला)

एस: ।

सी: हटाएं 2

सी: छोड़ो

S: +OK POP3 सर्वर साइनिंग ऑफ हो रहा है

उपयोगकर्ता एजेंट सबसे पहले मेल सर्वर से प्रत्येक संग्रहीत संदेश का आकार सूचीबद्ध करने के लिए कहता है। उपयोगकर्ता एजेंट तब सर्वर से प्रत्येक संदेश को पुनर्प्राप्त करता है और हटा देता है। ध्यान दें कि प्राधिकरण चरण के बाद,

उपयोगकर्ता एजेंट ने केवल चार कमांड का उपयोग किया: सूची , रिट्र , डेले , और quit . इनके लिए वाक्यविन्यास

कमांड को RFC 1939 में परिभाषित किया गया है। क्विट कमांड को संसाधित करने के बाद, POP3 सर्वर अपडेट चरण में प्रवेश करता है और मेलबॉक्स से संदेश 1 और 2 को हटा देता है।

इस डाउनलोड-और-डिलीट मोड के साथ एक समस्या यह है कि प्राप्तकर्ता, बॉब, खानाबदोश हो सकता है और
अपने मेल संदेशों को एकाधिक मशीनों से एक्सेस करना चाहते हैं, उदाहरण के लिए, उनके कार्यालय पीसी, उनके घर पीसी, और
उसका पोर्टेबल कंप्यूटर। डाउनलोड-और-डिलीट मोड बॉब के मेल संदेशों को इन तीन भागों में विभाजित करता है
मशीनें; खास तौर पर, अगर बॉब पहले अपने ऑफिस पीसी पर कोई संदेश पढ़ता है, तो वह शाम को घर पर अपने पोर्टेबल से संदेश को फिर से नहीं पढ़ पाएगा। डाउनलोड-एंड-कीप मोड में, उपयोगकर्ता एजेंट संदेशों को डाउनलोड करने के बाद उन्हें मेल सर्वर पर छोड़ देता है। इस मामले में, बॉब अलग-अलग मशीनों से संदेशों को फिर से पढ़ सकता है; वह काम से संदेश तक पहुँच सकता है और बाद में फिर से उस तक पहुँच सकता है

घर से एक सप्ताह दूर।

उपयोगकर्ता एजेंट और मेल सर्वर के बीच POP3 सत्र के दौरान, POP3 सर्वर कुछ स्थिति संबंधी जानकारी बनाए रखता है; विशेष रूप से, यह इस बात पर नज़र रखता है कि कौन से उपयोगकर्ता संदेशों को हटा दिया गया है।
हालाँकि, POP3 सर्वर POP3 सत्रों में स्टेट की जानकारी नहीं ले जाता है। सत्रों में स्टेट की जानकारी की यह कमी POP3 सर्वर के कार्यान्वयन को बहुत सरल बनाती है।

आईएमएपी

POP3 एक्सेस के साथ, एक बार बॉब अपने संदेशों को स्थानीय मशीन पर डाउनलोड कर लेता है, तो वह मेल बना सकता है

फ़ोल्डर्स और डाउनलोड किए गए संदेशों को फ़ोल्डर्स में ले जाएँ। बॉब फिर संदेशों को हटा सकता है, फ़ोल्डर्स में संदेशों को स्थानांतरित कर सकता है, और संदेशों की खोज कर सकता है (प्रेषक के नाम या विषय के अनुसार)। लेकिन यह प्रतिमान - अर्थात्, स्थानीय मशीन में फ़ोल्डर्स और संदेश - खानाबदोश उपयोगकर्ता के लिए एक समस्या पैदा करता है, जो किसी भी कंप्यूटर से एक्सेस किए जा सकने वाले रिमोट सर्वर पर फ़ोल्डर पदानुक्रम बनाए रखना पसंद करेगा। POP3 के साथ यह संभव नहीं है - POP3 प्रोटोकॉल उपयोगकर्ता को रिमोट फ़ोल्डर्स बनाने और फ़ोल्डर्स को संदेश असाइन करने का कोई साधन प्रदान नहीं करता है।

इस और अन्य समस्याओं को हल करने के लिए, [RFC 3501] में परिभाषित IMAP प्रोटोकॉल का आविष्कार किया गया था। POP3 की तरह, IMAP एक मेल एक्सेस प्रोटोकॉल है। इसमें POP3 की तुलना में कई ज़्यादा सुविधाएँ हैं, लेकिन यह काफी ज़्यादा जटिल भी है। (और इसलिए क्लाइंट और सर्वर साइड कार्यान्वयन काफी ज़्यादा जटिल हैं।)

IMAP सर्वर प्रत्येक संदेश को एक फ़ोल्डर से संबद्ध करेगा; जब कोई संदेश पहली बार सर्वर पर आता है, तो वह प्राप्तकर्ता के INBOX फ़ोल्डर से संबद्ध होता है। प्राप्तकर्ता तब संदेश को एक नए, उपयोगकर्ता द्वारा बनाए गए फ़ोल्डर में ले जा सकता है, संदेश को पढ़ सकता है, संदेश को हटा सकता है, और इसी तरह आगे बढ़ सकता है। IMAP प्रोटोकॉल उपयोगकर्ताओं को फ़ोल्डर बनाने और संदेशों को एक फ़ोल्डर से दूसरे फ़ोल्डर में ले जाने की अनुमति देने के लिए कमांड प्रदान करता है। IMAP ऐसे कमांड भी प्रदान करता है जो उपयोगकर्ताओं को विशिष्ट मानदंडों से मेल खाने वाले संदेशों के लिए दूरस्थ फ़ोल्डरों को खोजने की अनुमति देता है।

ध्यान दें कि, POP3 के विपरीत, IMAP सर्वर IMAP सत्रों में उपयोगकर्ता स्थिति की जानकारी बनाए रखता है - उदाहरण के लिए, फ़ोल्डरों के नाम और कौन से संदेश किस फ़ोल्डर से संबद्ध हैं।

IMAP की एक और महत्वपूर्ण विशेषता यह है कि इसमें ऐसे कमांड हैं जो उपयोगकर्ता एजेंट को संदेशों के घटक प्राप्त करने की अनुमति देते हैं। उदाहरण के लिए, उपयोगकर्ता एजेंट किसी संदेश का केवल संदेश शीर्षलेख या मल्टीपार्ट MIME संदेश का केवल एक भाग प्राप्त कर सकता है। यह सुविधा तब उपयोगी होती है जब उपयोगकर्ता एजेंट और उसके मेल सर्वर के बीच कम बैंडविड्थ कनेक्शन (उदाहरण के लिए, धीमी गति वाला मॉडेम लिंक) होता है। कम बैंडविड्थ कनेक्शन के साथ, उपयोगकर्ता अपने मेलबॉक्स में सभी संदेशों को डाउनलोड नहीं करना चाहेगा, विशेष रूप से लंबे संदेशों से बचना चाहिए जिनमें, उदाहरण के लिए, ऑडियो या वीडियो क्लिप हो सकती है।

वेब-आधारित ई-मेल

आज अधिक से अधिक उपयोगकर्ता अपने वेब ब्राउज़र के माध्यम से अपना ई-मेल भेज रहे हैं और उस तक पहुँच रहे हैं। हॉटमेल ने 1990 के दशक के मध्य में वेब-आधारित पहुँच की शुरुआत की। अब वेब-आधारित ई-मेल Google, Yahoo! के साथ-साथ लगभग हर प्रमुख विश्वविद्यालय और निगम द्वारा भी प्रदान किया जाता है। इस सेवा के साथ, उपयोगकर्ता एजेंट एक साधारण वेब ब्राउज़र है, और उपयोगकर्ता HTTP के माध्यम से अपने दूरस्थ मेलबॉक्स से संचार करता है। जब बॉब जैसे प्राप्तकर्ता अपने मेलबॉक्स में किसी संदेश तक पहुँचना चाहते हैं, तो ई-मेल संदेश बॉब के मेल सर्वर से बॉब के ब्राउज़र पर POP3 या IMAP प्रोटोकॉल के बजाय HTTP प्रोटोकॉल का उपयोग करके भेजा जाता है।

जब कोई प्रेषक, जैसे कि ऐलिस, ई-मेल संदेश भेजना चाहता है, तो ई-मेल संदेश उसके द्वारा भेजा जाता है
ब्राउज़र SMTP के बजाय HTTP के ज़रिए अपने मेल सर्वर पर भेजता है। हालाँकि, ऐलिस का मेल सर्वर अभी भी भेजता है
एसएमटीपी का उपयोग करके अन्य मेल सर्वरों को संदेश भेजता है और उनसे संदेश प्राप्त करता है।

2.4 DNS—इंटरनेट की निर्देशिका सेवा

हम इंसानों को कई तरीकों से पहचाना जा सकता है। उदाहरण के लिए, हमें हमारे जन्म प्रमाण पत्र पर लिखे नामों से पहचाना जा सकता है। हमें हमारे सामाजिक सुरक्षा नंबरों से पहचाना जा सकता है। हमें हमारे ड्राइविंग लाइसेंस नंबरों से पहचाना जा सकता है। हालाँकि इनमें से प्रत्येक पहचानकर्ता का उपयोग लोगों की पहचान करने के लिए किया जा सकता है, लेकिन किसी दिए गए संदर्भ में एक पहचानकर्ता दूसरे की तुलना में अधिक उपयुक्त हो सकता है। उदाहरण के लिए, IRS (संयुक्त राज्य अमेरिका में कुख्यात कर-संग्रह एजेंसी) के कंप्यूटर जन्म प्रमाण पत्र नामों के बजाय निश्चित-लंबाई वाले सामाजिक सुरक्षा नंबरों का उपयोग करना पसंद करते हैं। दूसरी ओर, आम लोग सामाजिक सुरक्षा नंबरों के बजाय अधिक स्मरणीय जन्म प्रमाण पत्र नामों को पसंद करते हैं। (वास्तव में, क्या आप यह कहने की कल्पना कर सकते हैं, "नमस्ते। मेरा नाम 132-67-9875 है। कृपया मेरे पति से मिलें, 178-87-1146।")

जैसे इंसानों को कई तरीकों से पहचाना जा सकता है, वैसे ही इंटरनेट होस्ट को भी पहचाना जा सकता है। होस्ट के लिए एक पहचानकर्ता उसका होस्टनाम है। होस्टनाम - जैसे www.facebook.com, www.google.com ,

gaia.cs.umass.edu —स्मृतिसूचक हैं और इसलिए मनुष्य द्वारा सराहे जाते हैं। हालाँकि, होस्टनाम इंटरनेट के भीतर होस्ट के स्थान के बारे में बहुत कम, यदि कोई हो, जानकारी प्रदान करते हैं। (ए होस्टनाम जैसे कि www.eurecom.fr , जो देश कोड .fr **के साथ समाप्त होता है** (संभवतः फ्रांस में, लेकिन , हमें बताता है कि मेज़बान है इससे अधिक कुछ नहीं कहा जा सकता है।) इसके अलावा, क्योंकि होस्टनाम में परिवर्तनीय-लंबाई वाले अल्फ़ान्यूमेरिक वर्ण शामिल हो सकते हैं, इसलिए उन्हें राउटर द्वारा संसाधित करना मुश्किल होगा। इन कारणों से, होस्ट को तथाकथित आईपी पते से भी पहचाना जाता है।

हमने अध्याय 4 में IP पतों के बारे में विस्तार से चर्चा की है , लेकिन अब उनके बारे में कुछ संक्षिप्त शब्द कहना उपयोगी होगा। एक IP पता चार बाइट्स का होता है और इसकी एक कठोर पदानुक्रमिक संरचना होती है। एक IP पता 121.7.106.83 जैसा दिखता है , जहाँ प्रत्येक अवधि दशमलव में व्यक्त बाइट्स में से एक को अलग करती है 0 से 255 तक का संकेतन। IP पता पदानुक्रमित होता है क्योंकि जब हम पते को बाएँ से दाएँ स्कैन करते हैं, तो हमें इस बारे में अधिक से अधिक विशिष्ट जानकारी मिलती है कि होस्ट इंटरनेट पर कहाँ स्थित है (यानी, किस नेटवर्क के भीतर, नेटवर्क के नेटवर्क में)। इसी तरह, जब हम डाक पते को नीचे से ऊपर की ओर स्कैन करते हैं, तो हमें इस बारे में अधिक से अधिक विशिष्ट जानकारी मिलती है कि पता प्राप्तकर्ता कहाँ स्थित है।

2.4.1 DNS द्वारा प्रदान की जाने वाली सेवाएँ

हमने अभी देखा कि होस्ट की पहचान करने के दो तरीके हैं - होस्टनाम द्वारा और आईपी एड्रेस द्वारा।
लोग ज़्यादा स्मरणीय होस्टनाम पहचानकर्ता को प्राथमिकता देते हैं, जबकि राउटर निश्चित-लंबाई, पदानुक्रमिक रूप से संरचित आईपी पते को प्राथमिकता देते हैं। इन प्राथमिकताओं को समेटने के लिए, हमें एक निर्देशिका सेवा की आवश्यकता है जो

होस्टनाम को IP पते में अनुवाद करता है। यह इंटरनेट के डोमेन नाम सिस्टम (DNS) का मुख्य कार्य है । **DNS** (1) DNS सर्वरों के पदानुक्रम में कार्यान्वित एक वितरित डेटाबेस है , और (2) एक

एप्लिकेशन-लेयर प्रोटोकॉल जो होस्ट को वितरित डेटाबेस को क्वेरी करने की अनुमति देता है। DNS सर्वर अक्सर

बर्कले इंटरनेट नेम डोमेन (BIND) सॉफ़्टवेयर चलाने वाली UNIX मशीनें [BIND 2016]। DNS प्रोटोकॉल UDP पर चलता है और पोर्ट 53 का उपयोग करता है।

DNS का उपयोग आम तौर पर अन्य एप्लिकेशन-लेयर प्रोटोकॉल द्वारा किया जाता है - जिसमें HTTP और SMTP शामिल हैं, ताकि उपयोगकर्ता द्वारा दिए गए होस्टनाम को IP पते में बदला जा सके। उदाहरण के लिए, विचार करें कि क्या होता है जब कोई ब्राउज़र (यानी, HTTP क्लाइंट), किसी उपयोगकर्ता के होस्ट पर चल रहा हो, URL का अनुरोध करता है

www.someschool.edu/index.html . उपयोगकर्ता के होस्ट को HTTP अनुरोध भेजने में सक्षम होने के लिए

वेब सर्वर www.someschool.edu **को संदेश** , उपयोगकर्ता के होस्ट को पहले उसका IP पता प्राप्त करना होगा

www.someschool.edu . यह निम्न प्रकार किया जाता है।

1. वही उपयोगकर्ता मशीन DNS अनुप्रयोग के क्लाइंट पक्ष को चलाती है।
2. ब्राउज़र DNS अनुप्रयोग के क्लाइंट साइड पर **होस्टनाम,** www.someschool.edu **होस्टनाम निकालता है।** , यूआरएल से और पास करता है
3. DNS क्लाइंट DNS सर्वर को होस्टनाम युक्त एक क्वेरी भेजता है।
4. DNS क्लाइंट को अंततः एक उत्तर प्राप्त होता है, जिसमें होस्टनाम के लिए IP पता शामिल होता है।
5. जब ब्राउज़र को DNS से IP पता प्राप्त हो जाता है, तो वह उस IP पते पर पोर्ट 80 पर स्थित HTTP सर्वर प्रक्रिया से TCP कनेक्शन आरंभ कर सकता है।

इस उदाहरण से हम देखते हैं कि DNS इसका उपयोग करने वाले इंटरनेट अनुप्रयोगों में एक अतिरिक्त विलंब जोड़ता है - कभी-कभी काफी अधिक। सौभाग्य से, जैसा कि हम नीचे चर्चा करते हैं, वांछित IP पता अक्सर "निकटवर्ती" DNS सर्वर में कैश किया जाता है, जो DNS नेटवर्क ट्रैफ़िक के साथ-साथ औसत DNS विलंब को कम करने में मदद करता है।

होस्टनामों को IP पतों में अनुवाद करने के अलावा DNS कुछ अन्य महत्वपूर्ण सेवाएं भी प्रदान करता है:

- होस्ट अलियासिंग। **जटिल होस्टनाम वाले होस्ट में एक या अधिक उपनाम नाम हो सकते हैं। उदाहरण के लिए,** relay1.west-coast.enterprise.com **जैसे** होस्टनाम में , मान लीजिए, दो उपनाम हो सकते हैं

  उपनाम जैसे कि enterprise.com और www.enterprise.com इस मामले में, होस्टनाम relay1.west-coast.enterprise.com **को एक** कैनोनिकल होस्टनाम कहा जाता है । उपनाम होस्टनाम, जब मौजूद होते हैं, तो आमतौर पर कैनोनिकल होस्टनाम की तुलना में अधिक स्मृतिवर्धक होते हैं। DNS को किसी एप्लिकेशन द्वारा आपूर्ति किए गए उपनाम होस्टनाम के साथ-साथ IP पते के लिए कैनोनिकल होस्टनाम प्राप्त करने के लिए आमंत्रित किया जा सकता है।

  मेज़बान का.

- मेल सर्वर एलियासिंग। स्पष्ट कारणों से, यह अत्यधिक वांछनीय है कि ई-मेल पते स्मृति-चिह्न वाले हों। उदाहरण के लिए, यदि बॉब का याहू मेल पर खाता है, तो बॉब का ई-मेल पता इतना सरल हो सकता है

  बॉब@याहू.मेल . हालाँकि, याहू मेल सर्वर का होस्टनाम yahoo.com **की तुलना में अधिक जटिल और बहुत कम स्मरणीय है।** (उदाहरण के लिए, विहित होस्टनाम relay1.west-coast.yahoo.com **जैसा कुछ हो सकता है**)। DNS को किसी मेल अनुप्रयोग द्वारा आपूर्ति किए गए उपनाम होस्टनाम के लिए विहित होस्टनाम तथा होस्ट का IP पता प्राप्त करने के लिए आमंत्रित किया जा सकता है।

  वास्तव में, MX रिकॉर्ड (नीचे देखें) किसी कंपनी के मेल सर्वर और वेब सर्वर को समान (उपनाम) होस्टनाम रखने की अनुमति देता है; उदाहरण के लिए, किसी कंपनी के वेब सर्वर और मेल सर्वर दोनों को ही MX कहा जा सकता है।

उद्यम.कॉम .

- लोड वितरण। DNS का उपयोग प्रतिरूपित सर्वरों के बीच लोड वितरण करने के लिए भी किया जाता है, जैसे प्रतिरूपित वेब सर्वर। व्यस्त साइटें, जैसे कि cnn.com , कई सर्वरों पर प्रतिकृति बनाई जाती है, प्रत्येक सर्वर अलग-अलग एंड सिस्टम पर चल रहा है और प्रत्येक का एक अलग आईपी पता है। प्रतिकृति के लिए वेब सर्वर, इस प्रकार IP पतों का एक सेट एक कैनोनिकल होस्टनाम से जुड़ा होता है। DNS डेटाबेस में IP पतों का यह सेट होता है। जब क्लाइंट पतों के एक सेट से मैप किए गए नाम के लिए DNS क्वेरी बनाते हैं, तो सर्वर IP पतों के पूरे सेट के साथ प्रतिक्रिया करता है, लेकिन प्रत्येक उत्तर के भीतर पतों के क्रम को घुमाता है। क्योंकि क्लाइंट आमतौर पर अपने HTTP अनुरोध संदेश को उस IP पते पर भेजता है जो सेट में सबसे पहले सूचीबद्ध होता है, DNS रोटेशन ट्रैफ़िक को प्रतिरूपित सर्वरों के बीच वितरित करता है। DNS रोटेशन का उपयोग ई-मेल के लिए भी किया जाता है ताकि कई मेल सर्वरों का एक ही उपनाम नाम हो सके। साथ ही, Akamai जैसी सामग्री वितरण कंपनियों ने DNS का उपयोग अधिक परिष्कृत तरीके से किया है

  वेब सामग्री वितरण प्रदान करने के तरीके [डिली 2002] (अनुभाग 2.6.3 देखें)।

DNS को RFC 1034 और RFC 1035 में निर्दिष्ट किया गया है, और कई अतिरिक्त RFC में अपडेट किया गया है। यह एक जटिल प्रणाली है, और हम इसके केवल मुख्य पहलुओं पर ही चर्चा करेंगे।

> व्यवहार में सिद्धांत
>
> DNS: क्लाइंट-सर्वर प्रतिमान के माध्यम से महत्वपूर्ण नेटवर्क फ़ंक्शन
>
> HTTP, FTP और SMTP की तरह, DNS प्रोटोकॉल एक एप्लीकेशन-लेयर प्रोटोकॉल है क्योंकि यह (1) क्लाइंट-सर्वर प्रतिमान का उपयोग करके संचार करने वाले एंड सिस्टम के बीच चलता है और (2) संचार करने वाले एंड सिस्टम के बीच DNS संदेशों को स्थानांतरित करने के लिए एक अंतर्निहित एंड-टू-एंड ट्रांसपोर्ट प्रोटोकॉल पर निर्भर करता है। हालाँकि, दूसरे अर्थ में, DNS की भूमिका वेब, फ़ाइल ट्रांसफ़र और ई-मेल एप्लिकेशन से काफी अलग है। इन एप्लिकेशन के विपरीत, DNS ऐसा एप्लिकेशन नहीं है जिसके साथ कोई उपयोगकर्ता सीधे इंटरैक्ट करता है। इसके बजाय, DNS एक मुख्य इंटरनेट फ़ंक्शन प्रदान करता है - अर्थात, उपयोगकर्ता एप्लिकेशन और अन्य सॉफ़्टवेयर के लिए होस्टनाम को उनके अंतर्निहित IP पतों में अनुवाद करना
>
> इंटरनेट में। हमने सेक्शन 1.2 में देखा कि इंटरनेट आर्किटेक्चर में बहुत सी जटिलताएँ नेटवर्क के "किनारों" पर स्थित हैं। DNS, जो नेटवर्क के किनारे पर स्थित क्लाइंट और सर्वर का उपयोग करके महत्वपूर्ण नाम-से-पता अनुवाद प्रक्रिया को लागू करता है, उस डिज़ाइन दर्शन का एक और उदाहरण है।

यहाँ ऑपरेशन। इच्छुक पाठक इन RFC और एल्बिट्ज़ और लियू [एल्बिट्ज़ 1993] की पुस्तक को संदर्भित करते हैं ; पूर्वव्यापी पेपर [मोकापेट्रिस 1988] भी देखें , जो DNS के क्या और क्यों का एक अच्छा विवरण प्रदान करता है, और [मोकापेट्रिस 2005]।

2.4.2 DNS कैसे काम करता है इसका अवलोकन

अब हम DNS के काम करने के तरीके का एक उच्च-स्तरीय अवलोकन प्रस्तुत करते हैं। हमारी चर्चा होस्टनाम-से-होस्टनाम पर केंद्रित होगी।

आईपी-पता अनुवाद सेवा.

मान लीजिए कि किसी उपयोगकर्ता के होस्ट में चल रहे किसी एप्लिकेशन (जैसे कि वेब ब्राउज़र या मेल रीडर) को होस्टनाम को IP पते में बदलने की आवश्यकता है। एप्लिकेशन DNS के क्लाइंट साइड को इनवोक करेगा, जो निर्दिष्ट करेगा

होस्टनाम जिसका अनुवाद किया जाना है। (कई UNIX-आधारित मशीनों पर, gethostbyname() वह फ़ंक्शन कॉल है जिसे अनुवाद करने के लिए कोई एप्लिकेशन कॉल करता है।) फिर उपयोगकर्ता के होस्ट में DNS नेटवर्क में क्वेरी संदेश भेजते हुए कार्यभार संभालता है। सभी DNS क्वेरी और उत्तर संदेश UDP डेटाग्राम के भीतर पोर्ट 53 पर भेजे जाते हैं। मिलीसेकंड से लेकर सेकंड तक की देरी के बाद, उपयोगकर्ता के होस्ट में DNS एक DNS उत्तर संदेश प्राप्त करता है जो वांछित मैपिंग प्रदान करता है। फिर यह मैपिंग इनवोकिंग एप्लिकेशन को पास की जाती है। इस प्रकार, उपयोगकर्ता के होस्ट में इनवोकिंग एप्लिकेशन के दृष्टिकोण से, DNS एक ब्लैक बॉक्स है जो एक सरल, सीधी अनुवाद सेवा प्रदान करता है। लेकिन वास्तव में, सेवा को लागू करने वाला ब्लैक बॉक्स जटिल है, जिसमें दुनिया भर में वितरित बड़ी संख्या में DNS सर्वर शामिल हैं, साथ ही एक एप्लिकेशन-लेयर प्रोटोकॉल भी है जो निर्दिष्ट करता है कि DNS सर्वर और क्वेरी करने वाले होस्ट कैसे काम करते हैं।

बातचीत करना।

DNS के लिए एक सरल डिज़ाइन में एक DNS सर्वर होगा जिसमें सभी मैपिंग शामिल होंगी। इस केंद्रीकृत डिज़ाइन में, क्लाइंट सभी क्वेरीज़ को एक ही DNS सर्वर पर निर्देशित करते हैं, और DNS सर्वर सीधे क्वेरी करने वाले क्लाइंट को जवाब देता है। हालाँकि इस डिज़ाइन की सादगी आकर्षक है, लेकिन यह आज के इंटरनेट के लिए अनुपयुक्त है, जिसमें होस्ट की विशाल (और बढ़ती) संख्या है। केंद्रीकृत डिज़ाइन की समस्याओं में शामिल हैं:

- विफलता का एक एकल बिंदु। यदि DNS सर्वर क्रैश हो जाता है, तो पूरा इंटरनेट भी क्रैश हो जाता है!
- ट्रैफ़िक वॉल्यूम। एक एकल DNS सर्वर को सभी DNS क्वेरीज़ (सैकड़ों मिलियन होस्ट्स से उत्पन्न सभी HTTP अनुरोधों और ई-मेल संदेशों के लिए) को संभालना होगा।
- दूर स्थित केंद्रीकृत डेटाबेस। एक एकल DNS सर्वर सभी क्वेरी करने वाले क्लाइंट के "निकट" नहीं हो सकता। यदि हम न्यूयॉर्क शहर में एकल DNS सर्वर लगाते हैं, तो ऑस्ट्रेलिया से सभी क्वेरीज़ को दुनिया के दूसरे छोर पर जाना होगा, शायद धीमे और भीड़भाड़ वाले लिंक के ज़रिए। इससे महत्वपूर्ण देरी हो सकती है।
- रखरखाव। एकल DNS सर्वर को सभी इंटरनेट होस्ट का रिकॉर्ड रखना होगा। न केवल यह केंद्रीकृत डेटाबेस बहुत बड़ा होगा, बल्कि इसे अक्सर अपडेट करना होगा।

  हर नये मेजबान.

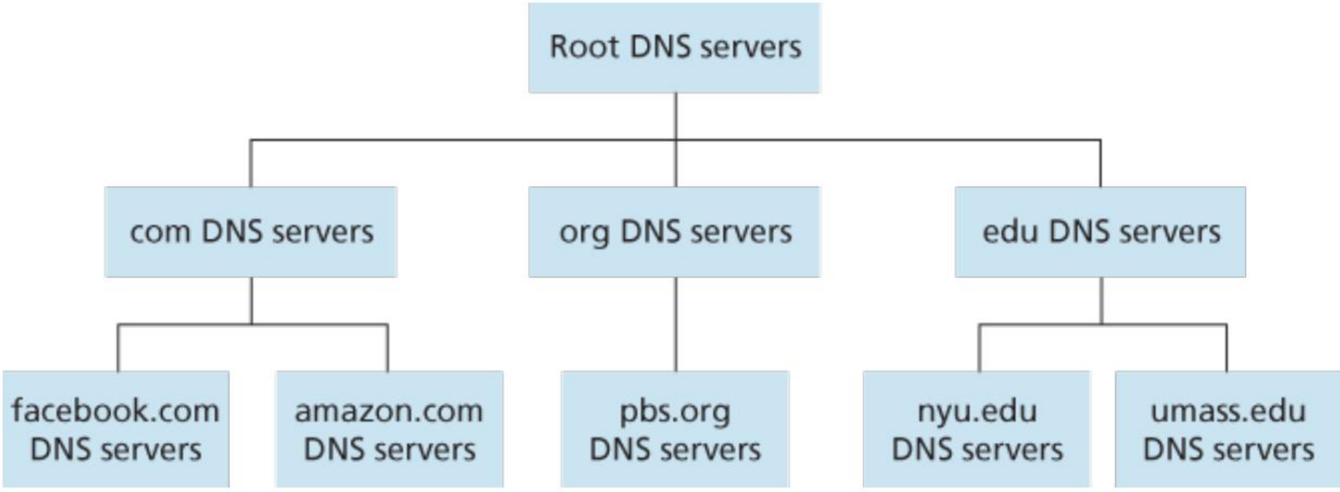
संक्षेप में, एक एकल DNS सर्वर में एक केंद्रीकृत डेटाबेस बस स्केल नहीं करता है। नतीजतन, DNS डिज़ाइन द्वारा वितरित किया जाता है। वास्तव में, DNS इस बात का एक अद्भुत उदाहरण है कि एक वितरित डेटाबेस कैसे हो सकता है
इंटरनेट पर लागू किया जाएगा।

एक वितरित, पदानुक्रमित डेटाबेस

पैमाने के मुद्दे से निपटने के लिए, DNS बड़ी संख्या में सर्वर का उपयोग करता है, जो पदानुक्रमिक तरीके से व्यवस्थित होते हैं और दुनिया भर में वितरित होते हैं।
किसी भी एक DNS सर्वर में इंटरनेट के सभी होस्ट के लिए सभी मैपिंग नहीं होती हैं। इसके बजाय, मैपिंग DNS सर्वर में वितरित की जाती हैं। पहले अनुमान के अनुसार, DNS सर्वर की तीन श्रेणियां हैं- रूट DNS सर्वर, शीर्ष-स्तरीय डोमेन (TLD) DNS

सर्वर, और आधिकारिक DNS सर्वर - चित्र 2.17 में दिखाए अनुसार पदानुक्रम में व्यवस्थित हैं । यह समझने के लिए कि सर्वर के ये तीन वर्ग कैसे परस्पर क्रिया करते हैं, मान लीजिए कि DNS क्लाइंट IP निर्धारित करना चाहता है

होस्टनाम का पता www.amazon.com . पहली बार



चित्र 2.17 DNS सर्वरों के पदानुक्रम का भाग

सन्निकटन के बाद, निम्नलिखित घटनाएँ घटित होंगी। क्लाइंट सबसे पहले रूट सर्वर में से किसी एक से संपर्क करता है,

जो शीर्ष-स्तरीय डोमेन com के लिए TLD सर्वर के लिए IP पते लौटाता है । क्लाइंट फिर एक से संपर्क करता है

इन TLD सर्वरों में से, जो amazon.com के लिए एक आधिकारिक सर्वर का IP पता लौटाता है । अंत में,

ग्राहक amazon.com के लिए आधिकारिक सर्वरों में से एक से संपर्क करता है , जो IP पता लौटाता है

होस्टनाम www.amazon.com हम जल्द ही इस DNS लुकअप प्रक्रिया की अधिक विस्तार से जांच करेंगे। लेकिन आइए पहले DNS सर्वर की इन तीन श्रेणियों पर करीब से नज़र डालें:

- रूट DNS सर्वर। पूरी दुनिया में 400 से ज़्यादा रूट नेम सर्वर फैले हुए हैं। चित्र 2.18 में वे देश दिखाए गए हैं, जिनके पास रूट नेम सर्वर हैं, जिनमें दस से ज़्यादा देश गहरे रंग में हैं। इन रूट नेम सर्वर को 13 अलग-अलग संगठनों द्वारा प्रबंधित किया जाता है। रूट नेम सर्वर की पूरी सूची, साथ ही उन्हें प्रबंधित करने वाले संगठन और उनके IP पते यहाँ देखे जा सकते हैं

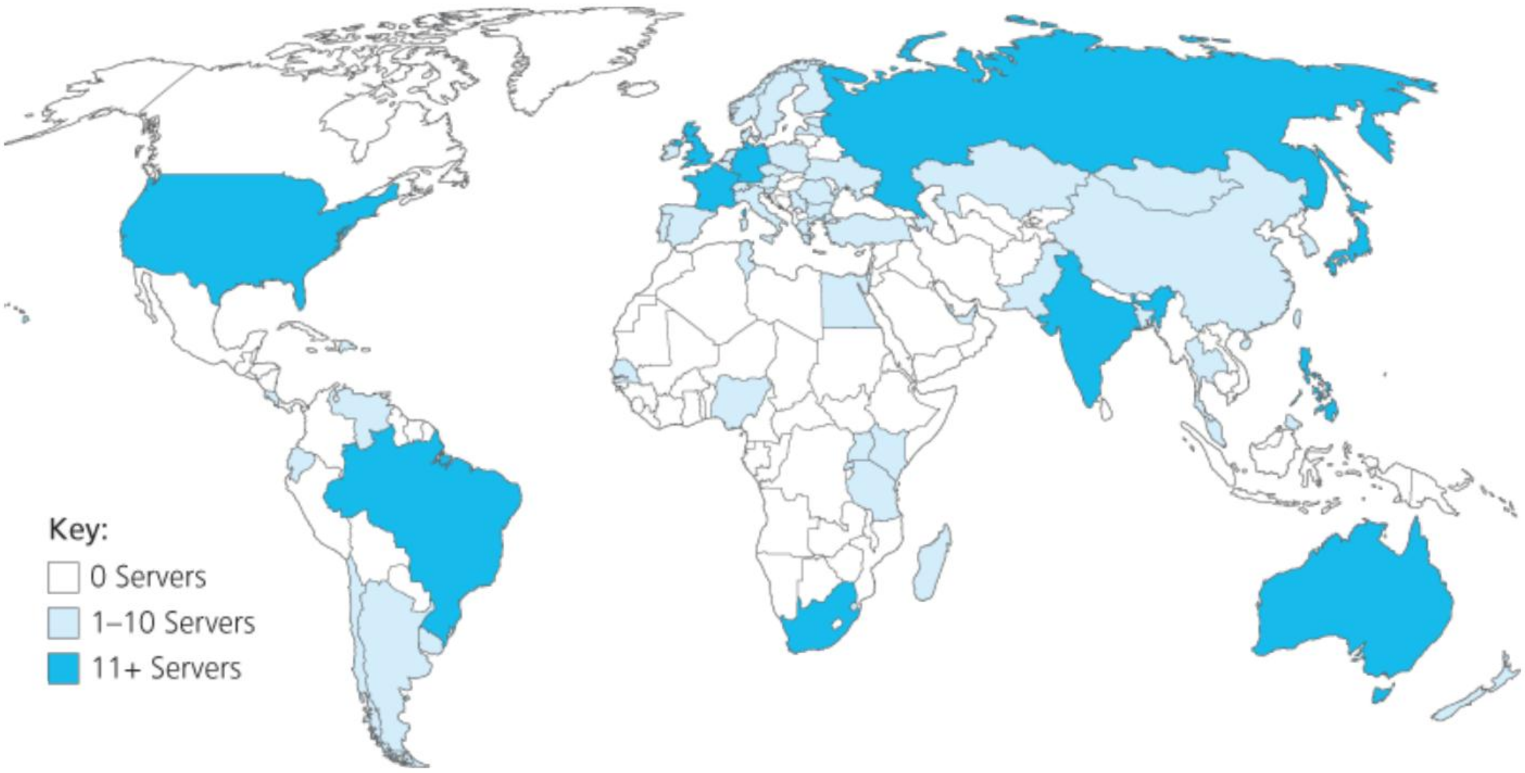
  [रूट सर्वर 2016] पर । रूट नाम सर्वर TLD सर्वर के IP पते प्रदान करते हैं।

- शीर्ष-स्तरीय डोमेन (TLD) सर्वर। प्रत्येक शीर्ष-स्तरीय डोमेन के लिए - शीर्ष-स्तरीय डोमेन जैसे com, org, net, edu, और gov, तथा सभी देश शीर्ष-स्तरीय डोमेन जैसे uk, fr, ca, और jp - एक TLD सर्वर (या सर्वर क्लस्टर) है। कंपनी Verisign Global Registry Services इसका रखरखाव करती है।

  com शीर्ष-स्तरीय डोमेन के लिए TLD सर्वर , और कंपनी Educause TLD का रखरखाव करती है

  edu शीर्ष-स्तरीय डोमेन के लिए सर्वर । TLD का समर्थन करने वाला नेटवर्क इंफ्रास्ट्रक्चर बड़ा और

  जटिल; वेरिसाइन नेटवर्क के अच्छे अवलोकन के लिए [ओस्टरवील 2012] देखें। सभी शीर्ष-स्तरीय डोमेन की सूची के लिए [TLD सूची 2016] देखें । TLD सर्वर आधिकारिक DNS सर्वर के लिए IP पते प्रदान करते हैं।



चित्र 2.18 2016 में DNS रूट सर्वर

- आधिकारिक DNS सर्वर। इंटरनेट पर सार्वजनिक रूप से सुलभ होस्ट (जैसे वेब सर्वर और मेल सर्वर) वाले प्रत्येक संगठन को सार्वजनिक रूप से सुलभ DNS रिकॉर्ड प्रदान करना चाहिए जो उन होस्ट के नामों को IP पतों से मैप करते हैं। किसी संगठन का आधिकारिक DNS सर्वर इन DNS रिकॉर्ड को रखता है। कोई संगठन इन रिकॉर्ड को रखने के लिए अपना खुद का आधिकारिक DNS सर्वर लागू करना चुन सकता है; वैकल्पिक रूप से, संगठन इन रिकॉर्ड को किसी सेवा प्रदाता के आधिकारिक DNS सर्वर में संग्रहीत करने के लिए भुगतान कर सकता है। अधिकांश विश्वविद्यालय और बड़ी कंपनियाँ अपने स्वयं के प्राथमिक और द्वितीयक (बैकअप) आधिकारिक DNS सर्वर को लागू और बनाए रखती हैं।

रूट, TLD और आधिकारिक DNS सर्वर सभी DNS सर्वर के पदानुक्रम से संबंधित हैं, जैसा कि चित्र 2.17 में दिखाया गया है। DNS सर्वर का एक और महत्वपूर्ण प्रकार है जिसे स्थानीय DNS सर्वर कहा जाता है। एक स्थानीय DNS सर्वर सख्ती से सर्वर के पदानुक्रम से संबंधित नहीं है, लेकिन फिर भी DNS आर्किटेक्चर के लिए केंद्रीय है। प्रत्येक ISP- जैसे कि एक आवासीय ISP या एक संस्थागत ISP- का एक स्थानीय DNS सर्वर होता है (जिसे डिफ़ॉल्ट नाम सर्वर भी कहा जाता है)। जब कोई होस्ट किसी ISP से जुड़ता है, तो ISP होस्ट को अपने एक या अधिक स्थानीय DNS सर्वरों के IP पते प्रदान करता है (आमतौर पर DHCP के माध्यम से, जिसकी चर्चा में किया गया है

अध्याय 4)। आप Windows या UNIX में नेटवर्क स्थिति विंडो तक पहुँच कर अपने स्थानीय DNS सर्वर का IP पता आसानी से निर्धारित कर सकते हैं। होस्ट का स्थानीय DNS सर्वर आमतौर पर होस्ट के "नज़दीक" होता है। संस्थागत ISP के लिए, स्थानीय DNS सर्वर होस्ट के समान LAN पर हो सकता है; आवासीय ISP के लिए, यह आमतौर पर होस्ट से कुछ राउटर से अलग होता है। जब कोई होस्ट DNS क्वेरी करता है, तो क्वेरी स्थानीय DNS सर्वर को भेजी जाती है, जो प्रॉक्सी का काम करता है, क्वेरी को DNS सर्वर पदानुक्रम में अग्रेषित करता है, जैसा कि हम नीचे और अधिक विस्तार से चर्चा करेंगे।

आइए एक सरल उदाहरण पर नज़र डालें। मान लीजिए होस्ट cse.nyu.edu को IP पता चाहिए

gaia.cs.umass.edu . **यह भी मान लीजिए कि** cse.nyu.edu के लिए NYU का स्थानीय DNS सर्वर कहा जाता है

dns.nyu.edu और gaia.cs.umass.edu के लिए एक आधिकारिक DNS सर्वर कहा जाता है

dns.umass.edu . जैसा कि चित्र 2.19 में दिखाया गया है, होस्ट cse.nyu.edu सबसे पहले एक DNS क्वेरी भेजता है

अपने स्थानीय DNS सर्वर, dns.nyu.edu को संदेश भेजें । क्वेरी संदेश में होस्टनाम शामिल है

अनुवादित, अर्थात्, gaia.cs.umass.edu स्थानीय DNS सर्वर क्वेरी संदेश को रूट DNS सर्वर पर अग्रेषित करता है। रूट DNS सर्वर edu प्रत्यय को नोट करता है और स्थानीय DNS सर्वर पर वापस लौटता है।

edu के लिए जिम्मेदार TLD सर्वर के लिए IP पते की सूची । स्थानीय DNS सर्वर फिर क्वेरी को फिर से भेजता है

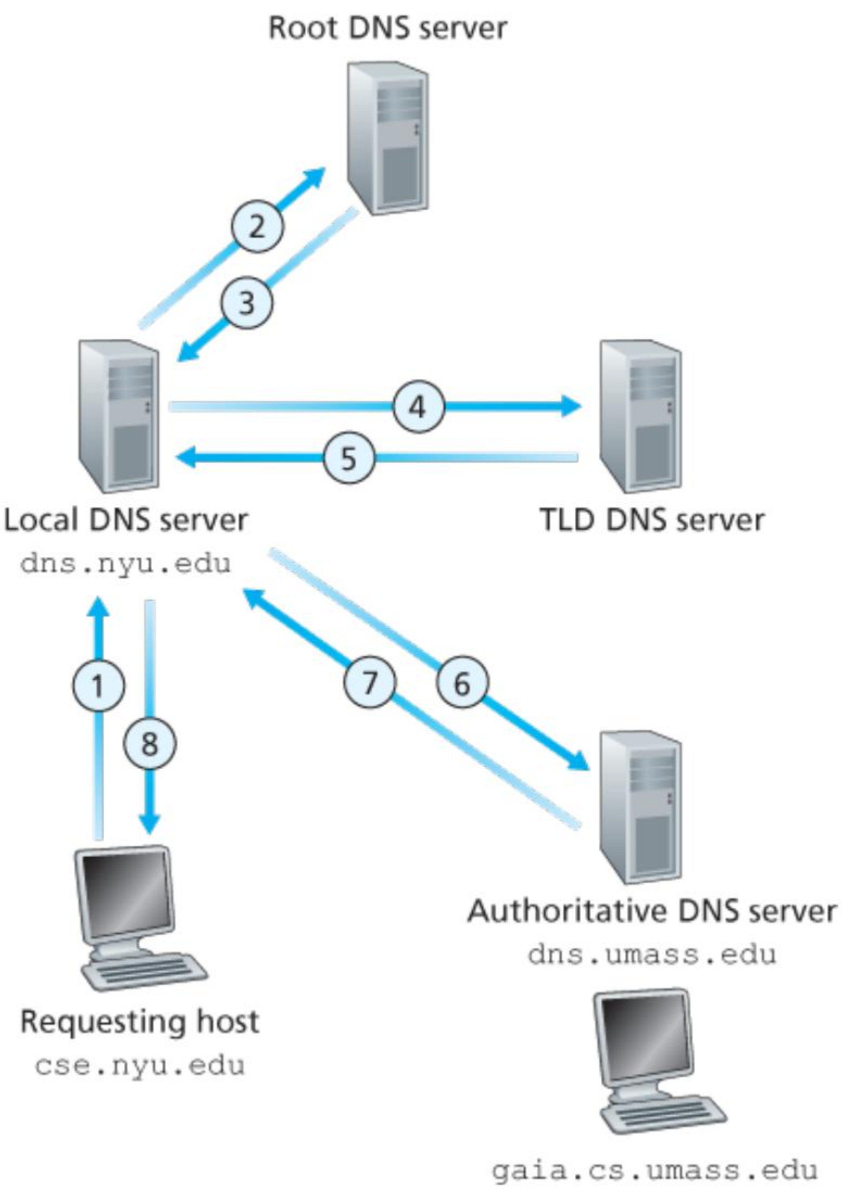
इनमें से किसी एक TLD सर्वर को संदेश। TLD सर्वर umass.edu पर ध्यान देता है प्रत्यय और मैसाचुसेट्स विश्वविद्यालय के लिए आधिकारिक DNS सर्वर के आईपी पते के साथ प्रतिक्रिया करता है,

अर्थात्, dns.umass.edu । अंत में, स्थानीय DNS सर्वर क्वेरी संदेश को सीधे पुनः भेजता है

dns.umass.edu , जो gaia.cs.umass.edu के आईपी पते के साथ प्रतिक्रिया करता है ध्यान दें कि इसमें

उदाहरण के लिए, एक होस्टनाम के लिए मैपिंग प्राप्त करने के लिए, आठ DNS संदेश भेजे गए: चार क्वेरी संदेश और चार उत्तर संदेश! हम जल्द ही देखेंगे कि DNS कैशिंग इस क्वेरी ट्रैफ़िक को कैसे कम करता है।

हमारे पिछले उदाहरण में यह माना गया था कि TLD सर्वर होस्टनाम के लिए आधिकारिक DNS सर्वर को जानता है। सामान्य तौर पर यह हमेशा सच नहीं होता। इसके बजाय, TLD सर्वर



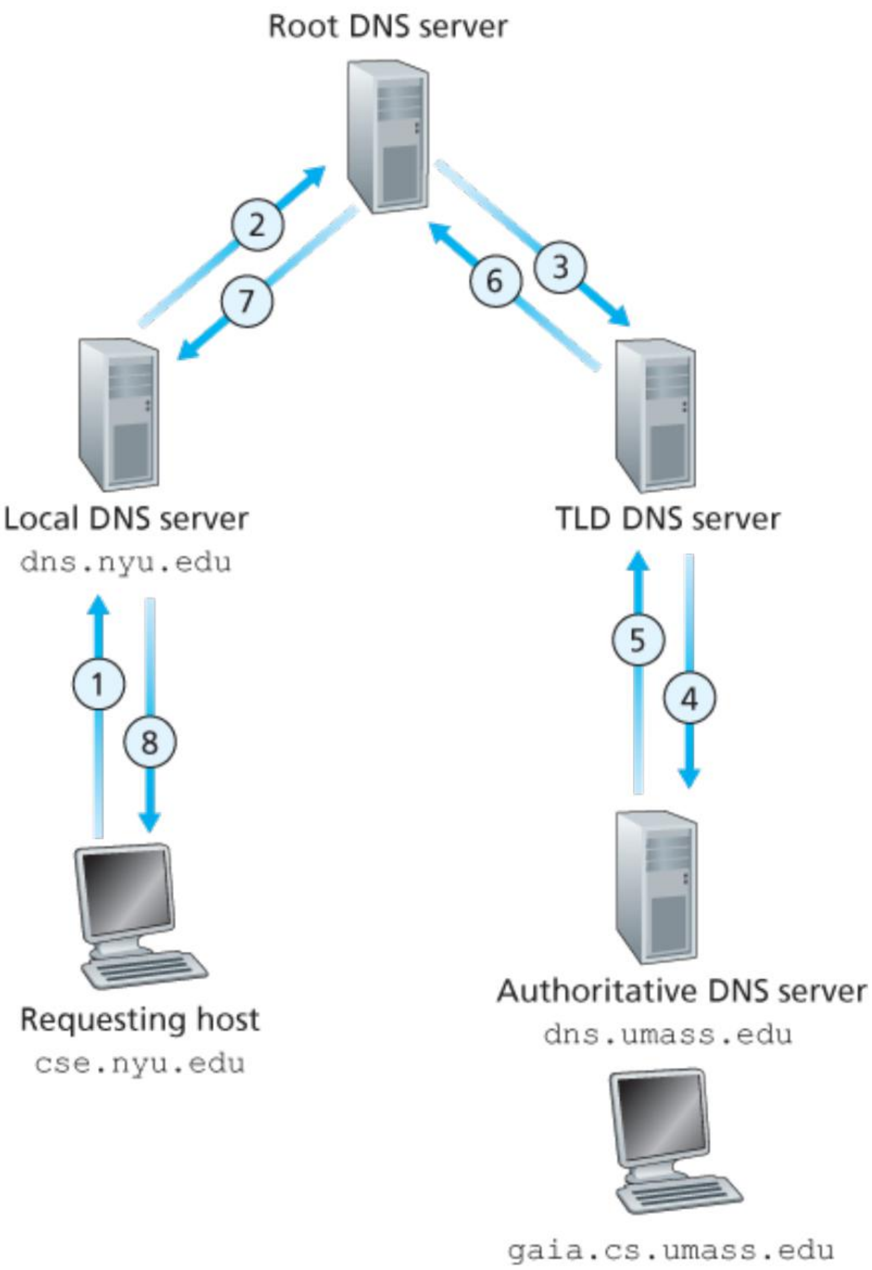
चित्र 2.19 विभिन्न DNS सर्वरों की परस्पर क्रिया

केवल एक मध्यवर्ती DNS सर्वर के बारे में ही पता हो सकता है, जो बदले में उस DNS सर्वर के लिए आधिकारिक DNS सर्वर को जानता है।
होस्टनाम। उदाहरण के लिए, फिर से मान लीजिए कि मैसाचुसेट्स विश्वविद्यालय के पास DNS सर्वर है
विश्वविद्यालय, जिसे dns.umass.edu कहा जाता है । यह भी मान लीजिए कि विश्वविद्यालय के प्रत्येक विभाग
मैसाचुसेट्स का अपना DNS सर्वर है, और प्रत्येक विभागीय DNS सर्वर सभी के लिए आधिकारिक है
विभाग में होस्ट करता है। इस मामले में, जब मध्यवर्ती DNS सर्वर, dns.umass.edu , प्राप्त करता है
cs.umass.edu पर समाप्त होने वाले होस्टनाम वाले होस्ट के लिए क्वेरी , यह dns.nyu.edu को IP लौटाता है
dns.cs.umass.edu का पता , जो cs.umass.edu पर समाप्त होने वाले सभी होस्टनामों के लिए आधिकारिक है .
स्थानीय DNS सर्वर dns.nyu.edu फिर क्वेरी को आधिकारिक DNS सर्वर को भेजता है, जो
स्थानीय DNS सर्वर को वांछित मैपिंग लौटाता है, जो बदले में अनुरोध करने वाले सर्वर को मैपिंग लौटाता है
इस स्थिति में, कुल 10 DNS संदेश भेजे जाते हैं!

चित्र 2.19 में दिखाया गया उदाहरण पुनरावर्ती क्वेरीज़ और पुनरावृत्तीय क्वेरीज़ दोनों का उपयोग करता है ।
cse.nyu.edu से dns.nyu.edu को भेजी गई क्वेरी एक पुनरावर्ती क्वेरी है, क्योंकि क्वेरी पूछती है
dns.nyu.edu को अपनी ओर से मैपिंग प्राप्त करने के लिए। लेकिन बाद की तीन क्वेरीज़ पुनरावृत्तीय हैं क्योंकि
सभी उत्तर सीधे dns.nyu.edu पर वापस आ जाते हैं । सिद्धांत रूप में, कोई भी DNS क्वेरी पुनरावृत्त या पुनरावृत्त हो सकती है
पुनरावर्ती। उदाहरण के लिए, चित्र 2.20 एक DNS क्वेरी श्रृंखला दिखाता है जिसके लिए सभी क्वेरी पुनरावर्ती हैं।
व्यवहार में, क्वेरीज़ आम तौर पर चित्र 2.19 में दिए गए पैटर्न का अनुसरण करती हैं: अनुरोधकर्ता होस्ट से क्वेरी
स्थानीय DNS सर्वर पुनरावर्ती है, और शेष क्वेरीज़ पुनरावृत्तीय हैं।

DNS कैशिंग

अब तक की हमारी चर्चा में DNS कैशिंग को नजरअंदाज किया गया है, जो DNS प्रणाली की एक अत्यंत महत्वपूर्ण विशेषता है।
सच तो यह है कि DNS विलंबित प्रदर्शन को बेहतर बनाने और विलंबित प्रदर्शन को कम करने के लिए DNS कैशिंग का बड़े पैमाने पर दोहन करता है।
DNS संदेशों की संख्या



चित्र 2.20 DNS में पुनरावर्ती क्वेरीज़

इंटरनेट पर चारों ओर घूम रहा है। DNS कैशिंग के पीछे का विचार बहुत सरल है। क्वेरी चेन में, जब DNS सर्वर को DNS उत्तर प्राप्त होता है (उदाहरण के लिए, होस्टनाम से IP तक मैपिंग शामिल है)

पता), यह मैपिंग को अपनी स्थानीय मेमोरी में कैश कर सकता है। उदाहरण के लिए, चित्र 2.19 में, हर बार जब स्थानीय DNS सर्वर dns.nyu.edu किसी DNS सर्वर से उत्तर प्राप्त करता है, तो यह उत्तर में मौजूद किसी भी जानकारी को कैश कर सकता है। यदि DNS सर्वर में होस्टनाम/IP पता जोड़ी कैश की गई है और उसी होस्टनाम के लिए DNS सर्वर पर कोई अन्य क्वेरी आती है, तो DNS सर्वर वांछित IP पता प्रदान कर सकता है, भले ही वह होस्टनाम के लिए आधिकारिक न हो। क्योंकि होस्टनाम और IP पतों के बीच होस्ट और मैपिंग किसी भी तरह से स्थायी नहीं हैं, DNS सर्वर एक समय अवधि (अक्सर दो दिन के लिए सेट) के बाद कैश की गई जानकारी को त्याग देते हैं।

उदाहरण के लिए, मान लीजिए कि एक होस्ट apricot.nyu.edu , dns.nyu.edu से IP पता पूछता है ।

होस्टनाम cnn.com इसके अलावा, मान लीजिए कि कुछ घंटों बाद, एक और NYU होस्ट, मान लीजिए,

उसी होस्टनाम के साथ dns.nyu.edu को भी क्वेरी करता है । **कैशिंग के कारण, स्थानीय** kiwi.nyu.edu ,

DNS सर्वर cnn.com का IP पता तुरंत लौटाने में सक्षम होगा इस दूसरे अनुरोध के लिए

होस्ट को किसी अन्य DNS सर्वर से क्वेरी किए बिना। एक स्थानीय DNS सर्वर TLD सर्वर के IP पते को भी कैश कर सकता है, जिससे स्थानीय DNS सर्वर को क्वेरी चेन में रूट DNS सर्वर को बायपास करने की अनुमति मिलती है। वास्तव में, कैशिंग के कारण, DNS क्वेरी के एक बहुत छोटे हिस्से को छोड़कर सभी के लिए रूट सर्वर को बायपास किया जाता है।

2.4.3 DNS रिकॉर्ड और संदेश

DNS सर्वर जो मिलकर DNS वितरित डेटाबेस को क्रियान्वित करते हैं, संसाधन रिकॉर्ड (RR) संग्रहीत करते हैं, जिसमें होस्टनाम-से-IP पता मैपिंग प्रदान करने वाले RR शामिल हैं। प्रत्येक DNS उत्तर संदेश में एक या अधिक संसाधन रिकॉर्ड होते हैं। इस और अगले उपखंड में, हम DNS का संक्षिप्त अवलोकन प्रदान करते हैं

संसाधन रिकॉर्ड और संदेश; अधिक विवरण [अल्बिट्ज़ 1993] या DNS RFCs [RFC 1034; RFC 1035] में पाया जा सकता है।

संसाधन रिकॉर्ड एक चार-टपल है जिसमें निम्नलिखित फ़ील्ड शामिल हैं:

```
(नाम, मूल्य, प्रकार, टीटीएल)
```

टीटीएल संसाधन रिकॉर्ड का जीवित रहने का समय है; यह निर्धारित करता है कि किसी संसाधन को कब हटाया जाना चाहिए कैश। नीचे दिए गए उदाहरण रिकॉर्ड में, हम TTL **फ़ील्ड को अनदेखा करते हैं।** नाम और मान का अर्थ प्रकार पर निर्भर :

- यदि Type=A है , तो नाम एक होस्टनाम है और Value होस्टनाम के लिए IP पता है। इस प्रकार, एक Type A रिकॉर्ड मानक होस्टनाम-से-IP पता मैपिंग प्रदान करता है। उदाहरण के लिए,

  (relay1.bar.foo.com, 145.37.93.126, A) एक प्रकार A रिकॉर्ड है.
- यदि Type=NS है , तो Name एक डोमेन है (जैसे foo.com) और Value एक होस्टनाम है आधिकारिक DNS सर्वर जो डोमेन में होस्ट के लिए IP पते प्राप्त करना जानता है। यह रिकॉर्ड का उपयोग क्वेरी श्रृंखला में DNS क्वेरी को आगे रूट करने के लिए किया जाता है। उदाहरण के लिए, (foo.com, dns.foo.com, NS) एक प्रकार का NS रिकॉर्ड है।
- यदि Type=CNAME है , तो Value उपनाम होस्टनाम Name **के लिए एक विहित होस्टनाम है** । यह रिकॉर्ड क्वेरी करने वाले होस्ट को होस्टनाम के लिए विहित नाम प्रदान कर सकता है। उदाहरण के लिए, (foo.com, relay1.bar.foo.com, CNAME) एक CNAME रिकॉर्ड है.
- यदि Type=MX है , तो Value उस मेल सर्वर का विहित नाम है जिसका उपनाम होस्टनाम Name **है** । उदाहरण के लिए, (foo.com, mail.bar.foo.com, MX) एक MX रिकॉर्ड है। MX रिकॉर्ड की अनुमति देता है मेल सर्वर के होस्टनामों को सरल उपनामों से युक्त करें। ध्यान दें कि MX रिकॉर्ड का उपयोग करके, कोई कंपनी अपने मेल सर्वर और अपने किसी अन्य सर्वर (जैसे कि उसका वेब सर्वर) के लिए समान उपनाम वाला नाम रख सकती है। मेल सर्वर के लिए विहित नाम प्राप्त करने के लिए, DNS क्लाइंट MX के लिए क्वेरी करेगा

रिकॉर्ड; दूसरे सर्वर के लिए विहित नाम प्राप्त करने के लिए, DNS क्लाइंट CNAME के लिए क्वेरी करेगा
अभिलेख।

यदि DNS सर्वर किसी विशेष होस्टनाम के लिए आधिकारिक है, तो DNS सर्वर में होस्टनाम के लिए एक प्रकार A रिकॉर्ड होगा। (यदि DNS सर्वर आधिकारिक नहीं भी है, तो भी इसके कैश में एक प्रकार A रिकॉर्ड हो सकता है।) यदि सर्वर किसी होस्टनाम के लिए आधिकारिक नहीं है, तो सर्वर में उस डोमेन के लिए एक प्रकार NS रिकॉर्ड होगा जिसमें होस्टनाम शामिल है; इसमें एक प्रकार A रिकॉर्ड भी होगा जो IP प्रदान करता है

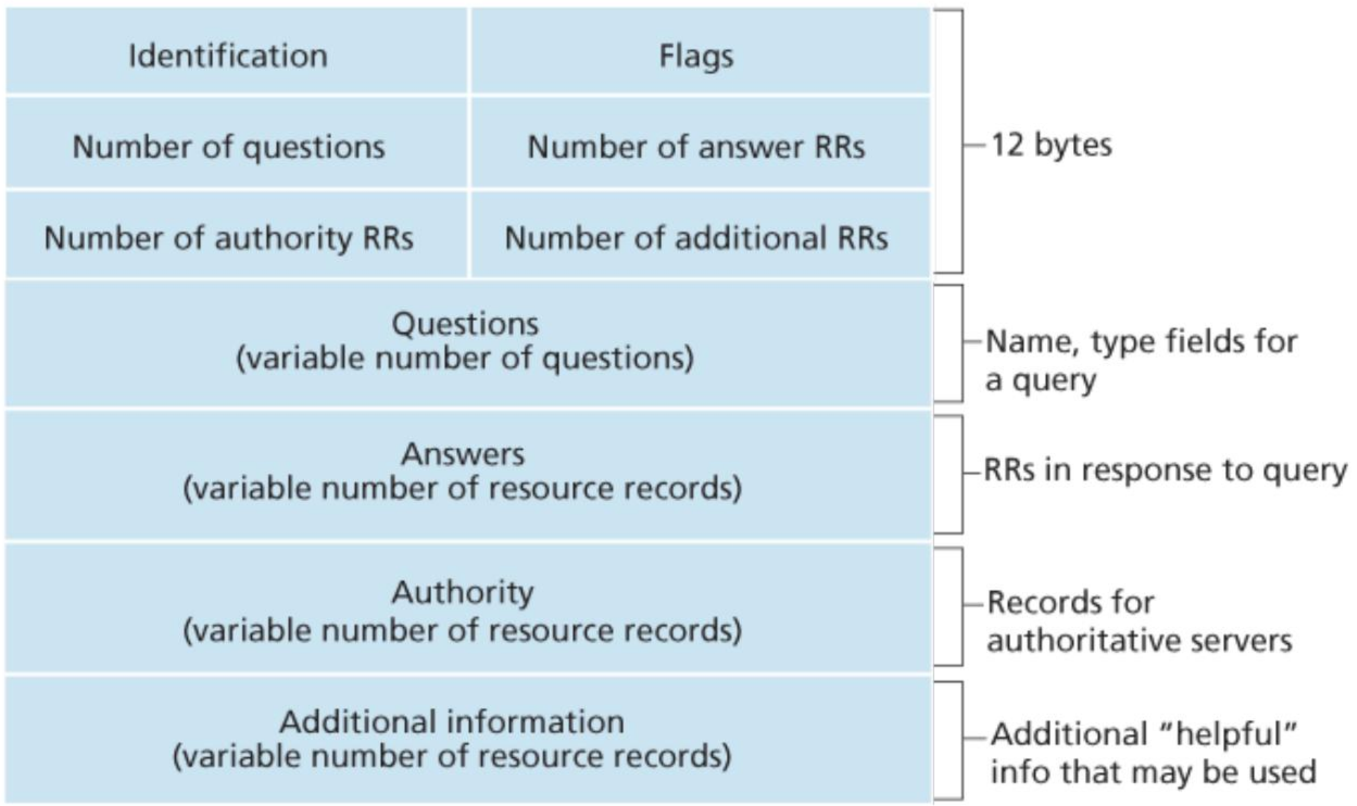
NS रिकॉर्ड के Value फ़ील्ड में DNS सर्वर का पता । **उदाहरण के लिए, मान लीजिए कि** edu TLD सर्वर होस्ट gaia.cs.umass.edu **के लिए अधिकृत नहीं है** . तो इस सर्वर में एक रिकॉर्ड होगा एक डोमेन जिसमें होस्ट gaia.cs.umass.edu **शामिल है** उदाहरण के लिए, (umass.edu, dns.umass.edu, NS) I edu TLD सर्वर में एक टाइप A रिकॉर्ड भी होगा, जो मैप करता है DNS सर्वर dns.umass.edu को एक आईपी पते पर, उदाहरण के लिए, (dns.umass.edu, 128.119.40.111, ए) .

डीएनएस संदेश

इस खंड में पहले हमने DNS क्वेरी और उत्तर संदेशों का उल्लेख किया था। ये DNS संदेशों के केवल दो प्रकार हैं। इसके अलावा, क्वेरी और उत्तर संदेश दोनों का प्रारूप समान है, जैसा कि नीचे दिखाया गया है

चित्र 2.21. DNS संदेश में विभिन्न फ़ील्डों का अर्थ निम्नानुसार है:

- पहले 12 बाइट्स हेडर सेक्शन हैं, जिसमें कई फ़ील्ड हैं। पहला फ़ील्ड 16-बिट नंबर है जो क्वेरी की पहचान करता है। इस पहचानकर्ता को क्वेरी के उत्तर संदेश में कॉपी किया जाता है, जिससे क्लाइंट को प्राप्त उत्तरों को भेजी गई क्वेरी से मिलान करने की अनुमति मिलती है। फ़्लैग फ़ील्ड में कई फ़्लैग हैं। 1-बिट क्वेरी/रिप्लाई फ़्लैग यह दर्शाता है कि संदेश क्वेरी (0) है या उत्तर (1)। 1-बिट आधिकारिक फ़्लैग है



चित्र 2.21 DNS संदेश प्रारूप

जब DNS सर्वर किसी क्वेरी किए गए नाम के लिए आधिकारिक सर्वर होता है, तो उत्तर संदेश में सेट किया जाता है। 1-बिट रिकर्सन-वांछित ध्वज तब सेट किया जाता है जब कोई क्लाइंट (होस्ट या DNS सर्वर) चाहता है कि DNS सर्वर रिकर्सन करे, जब उसके पास रिकॉर्ड न हो। यदि DNS सर्वर रिकर्सन का समर्थन करता है, तो उत्तर में 1-बिट रिकर्सन-उपलब्ध फ़ील्ड सेट किया जाता है। हेडर में, चार संख्या-फ़ील्ड भी हैं। ये फ़ील्ड हेडर के बाद आने वाले चार प्रकार के डेटा अनुभागों की घटनाओं की संख्या को इंगित करते हैं।

- प्रश्न अनुभाग में उस क्वेरी के बारे में जानकारी होती है जो की जा रही है। इस अनुभाग में (1) एक नाम फ़ील्ड शामिल है जिसमें वह नाम होता है जिसके बारे में पूछा जा रहा है, और (2) एक प्रकार फ़ील्ड जो नाम के बारे में पूछे जा रहे प्रश्न के प्रकार को इंगित करता है - उदाहरण के लिए, नाम से जुड़ा होस्ट पता (प्रकार A) या नाम के लिए मेल सर्वर (प्रकार MX)।

- DNS सर्वर से प्राप्त उत्तर में, उत्तर अनुभाग में उस नाम के लिए संसाधन रिकॉर्ड होते हैं, जिसके बारे में मूल रूप से पूछा गया था। याद रखें कि प्रत्येक संसाधन रिकॉर्ड में प्रकार (उदाहरण के लिए, A, NS,
CNAME और MX) के लिए, मान होस्टनाम में , और TTL । एक उत्तर उत्तर में कई RR लौटा सकता है, क्योंकि
एकाधिक IP पते हो सकते हैं (उदाहरण के लिए, प्रतिरूपित वेब सर्वर के लिए, जैसा कि इस अनुभाग में पहले चर्चा की गई है)।

- प्राधिकरण अनुभाग में अन्य आधिकारिक सर्वरों के रिकॉर्ड होते हैं।
- अतिरिक्त अनुभाग में अन्य सहायक रिकॉर्ड शामिल हैं। उदाहरण के लिए, MX क्वेरी के उत्तर में उत्तर फ़ील्ड में एक संसाधन रिकॉर्ड होता है जो मेल सर्वर का कैनोनिकल होस्टनाम प्रदान करता है। अतिरिक्त अनुभाग में एक प्रकार A रिकॉर्ड होता है जो किसी मेल सर्वर के कैनोनिकल होस्टनाम के लिए IP पता प्रदान करता है।
मेल सर्वर.

आप जिस होस्ट पर काम कर रहे हैं, उससे सीधे किसी DNS सर्वर पर DNS क्वेरी संदेश कैसे भेजना चाहेंगे? यह nslookup प्रोग्राम के साथ आसानी से किया जा सकता है, जो कि अधिकांश Windows और UNIX प्लेटफ़ॉर्म से उपलब्ध है। उदाहरण के लिए, Windows होस्ट से, कमांड प्रॉम्प्ट खोलें और बस "nslookup" टाइप करके nslookup प्रोग्राम को इनवोक करें। nslookup को इनवोक करने के बाद, आप किसी भी DNS सर्वर (रूट, TLD या आधिकारिक) को DNS क्वेरी भेज सकते हैं। DNS सर्वर से उत्तर संदेश प्राप्त करने के बाद, nslookup उत्तर में शामिल रिकॉर्ड (मानव-पठनीय प्रारूप में) प्रदर्शित करेगा। अपने स्वयं के होस्ट से nslookup चलाने के विकल्प के रूप में, आप कई वेब साइटों में से एक पर जा सकते हैं जो आपको दूरस्थ रूप से nslookup का उपयोग करने की अनुमति देती हैं। (बस एक खोज इंजन में "nslookup" टाइप करें और आपको इनमें से किसी एक साइट पर ले जाया जाएगा।) इस अध्याय के अंत में DNS Wireshark लैब आपको DNS को बहुत अधिक तरीके से एक्सप्लोर करने की अनुमति देगा।

और अधिक विस्तृत जानकारी।

DNS डेटाबेस में रिकॉर्ड सम्मिलित करना

ऊपर चर्चा इस बात पर केंद्रित थी कि DNS डेटाबेस से रिकॉर्ड कैसे प्राप्त किए जाते हैं। आप सोच रहे होंगे कि रिकॉर्ड सबसे पहले डेटाबेस में कैसे आते हैं। आइए एक विशिष्ट उदाहरण के संदर्भ में देखें कि यह कैसे किया जाता है। मान लीजिए कि आपने अभी-अभी नेटवर्क यूटोपिया नामक एक रोमांचक नई स्टार्टअप कंपनी बनाई है। पहली चीज़ जो आप निश्चित रूप से करना चाहेंगे वह है डोमेन नाम पंजीकृत करना

networkutopia.com रजिस्ट्रार के पास। रजिस्ट्रार एक वाणिज्यिक इकाई है जो डोमेन नाम की विशिष्टता की पुष्टि करता है, डोमेन नाम को DNS डेटाबेस में दर्ज करता है (जैसा कि नीचे चर्चा की गई है), और अपनी सेवाओं के लिए आपसे एक छोटा सा शुल्क वसूलता है। 1999 से पहले, एक एकल रजिस्ट्रार, नेटवर्क सॉल्यूशंस का एकाधिकार था

कॉम के लिए डोमेन नाम पंजीकरण पर ग्राहकों के लिए , जाल , और org डोमेन। लेकिन अब कई रजिस्ट्रार हैं प्रतिस्पर्धा, और इंटरनेट कॉर्पोरेशन फॉर असाइन्ड नेम्स एंड नंबर्स (आईसीएएनएन) विभिन्न रजिस्ट्रारों को मान्यता प्रदान करता है। मान्यता प्राप्त रजिस्ट्रारों की पूरी सूची http:// पर उपलब्ध है www.internic.net .

जब आप डोमेन नाम networkutopia.com पंजीकृत करते हैं कुछ रजिस्ट्रार के साथ, आपको रजिस्ट्रार को अपने प्राथमिक और द्वितीयक आधिकारिक DNS के नाम और आईपी पते भी प्रदान करने होंगे सर्वर। मान लीजिए नाम और आईपी पते dns1.networkutopia.com हैं , dns2.networkutopia.com , 212.2.212.1, और 212.212.212.2। इन दो आधिकारिक DNS सर्वरों में से प्रत्येक के लिए, रजिस्ट्रार तब सुनिश्चित करेगा कि टाइप NS और टाइप A रिकॉर्ड TLD com सर्वर में दर्ज किए गए हैं। विशेष रूप से, प्राथमिक आधिकारिक सर्वर के लिए

रजिस्ट्रार DNS networkutopia.com में निम्नलिखित दो संसाधन रिकॉर्ड सम्मिलित करेगा , प्रणाली:

(networkutopia.com, dns1.networkutopia.com, एनएस)
(dns1.networkutopia.com, 212.212.212.1, ए)

आपको यह भी सुनिश्चित करना होगा कि आपके वेब सर्वर के लिए टाइप A संसाधन रिकॉर्ड www.networkutopia.com और आपके मेल सर्वर के लिए टाइप MX संसाधन रिकॉर्ड mail.networkutopia.com आपके आधिकारिक DNS में दर्ज किए गए हैं

सुरक्षा पर ध्यान दें

DNS कमज़ोरियाँ

हमने देखा है कि DNS इंटरनेट अवसंरचना का एक महत्वपूर्ण घटक है, तथा कई महत्वपूर्ण सेवाएं - जिनमें वेब और ई-मेल भी शामिल हैं - इसके बिना काम करने में असमर्थ हैं।

इसलिए स्वाभाविक रूप से हम पूछते हैं, DNS पर हमला कैसे किया जा सकता है? क्या DNS एक ऐसी जगह है, जो सेवा से बाहर होने का इंतज़ार कर रही है, और साथ ही साथ अधिकांश इंटरनेट एप्लिकेशन को भी अपने साथ ले जा रही है?

दिमाग में आने वाला पहला हमला DNS सर्वर के खिलाफ DDoS बैंडविड्थ-फ्लडिंग हमला है (अनुभाग 1.6 देखें) । उदाहरण के लिए, एक हमलावर प्रत्येक DNS रूट सर्वर को पैकेटों की बाढ़ भेजने का प्रयास कर सकता है, इतने सारे कि अधिकांश वैध DNS क्वेरीज़ का कभी जवाब नहीं मिलता। DNS रूट सर्वर के खिलाफ़ इस तरह का एक बड़े पैमाने पर DDoS हमला वास्तव में 21 अक्टूबर, 2002 को हुआ था। इस हमले में, हमलावरों ने 13 DNS रूट IP पतों में से प्रत्येक को ICMP पिंग संदेशों का ट्रक लोड भेजने के लिए एक बॉटनेट का लाभ उठाया। (ICMP संदेशों पर चर्चा की गई है

धारा 5.6. अभी के लिए, यह जानना पर्याप्त है कि ICMP पैकेट विशेष प्रकार के IP डेटाग्राम हैं।)
सौभाग्य से, इस बड़े पैमाने पर हमले से न्यूनतम क्षति हुई, जिसका उपयोगकर्ताओं के इंटरनेट अनुभव पर बहुत कम या कोई प्रभाव नहीं पड़ा। हमलावर रूट सर्वर पर पैकेट की बाढ़ को निर्देशित करने में सफल रहे। लेकिन कई DNS रूट सर्वर पैकेट फ़िल्टर द्वारा संरक्षित थे, जो रूट सर्वर पर निर्देशित सभी ICMP पिंग संदेशों को हमेशा ब्लॉक करने के लिए कॉन्फ़िगर किए गए थे। इस प्रकार ये संरक्षित सर्वर बच गए और सामान्य रूप से काम करने लगे। इसके अलावा, अधिकांश स्थानीय DNS सर्वर शीर्ष-स्तरीय-डोमेन सर्वर के IP पते को कैश करते हैं, जिससे क्वेरी प्रक्रिया अक्सर DNS रूट सर्वर को बायपास कर सकती है।

DNS के विरुद्ध संभावित रूप से अधिक प्रभावी DDoS हमला शीर्ष-स्तरीय-डोमेन सर्वरों पर DNS क्वेरीज़ की बाढ़ भेजना होगा, उदाहरण के लिए, सभी शीर्ष-स्तरीय-डोमेन सर्वरों पर जो .com डोमेन को संभालते हैं। DNS सर्वरों को निर्देशित DNS क्वेरीज़ को फ़िल्टर करना कठिन होगा; और शीर्ष-स्तरीय-डोमेन सर्वर रूट सर्वर की तरह आसानी से बायपास नहीं किए जा सकते हैं। लेकिन स्थानीय DNS सर्वरों में कैशिंग करके इस तरह के हमले की गंभीरता को आंशिक रूप से कम किया जा सकता है।

DNS पर संभावित रूप से अन्य तरीकों से हमला किया जा सकता है। मैन-इन-द-मिडल अटैक में, हमलावर होस्ट से क्वेरी को इंटरसेप्ट करता है और फर्जी जवाब देता है। DNS पॉइज़निंग अटैक में, हमलावर DNS सर्वर को फर्जी जवाब भेजता है, जिससे सर्वर अपने कैश में फर्जी रिकॉर्ड स्वीकार करने के लिए धोखा देता है। इनमें से किसी भी हमले का इस्तेमाल, उदाहरण के लिए, किसी अनजान वेब उपयोगकर्ता को हमलावर की वेब साइट पर रीडायरेक्ट करने के लिए किया जा सकता है। हालाँकि, इन हमलों को लागू करना मुश्किल है, क्योंकि वे

पैकेटों को रोकने या सर्वर को थ्रॉटलिंग करने की आवश्यकता होती है [स्कोडिस 2006]।

संक्षेप में, DNS ने हमलों के खिलाफ़ खुद को आश्चर्यजनक रूप से मज़बूत साबित किया है। आज तक, ऐसा कोई हमला नहीं हुआ है जिसने DNS सेवा को सफलतापूर्वक बाधित किया हो।

सर्वर। (हाल ही तक, प्रत्येक DNS सर्वर की सामग्री स्थिर रूप से कॉन्फ़िगर की गई थी, उदाहरण के लिए, सिस्टम मैनेजर द्वारा बनाई गई कॉन्फ़िगरेशन फ़ाइल से। हाल ही में, DNS प्रोटोकॉल में एक अपडेट विकल्प जोड़ा गया है ताकि DNS के माध्यम से डेटाबेस से डेटा को गतिशील रूप से जोड़ा या हटाया जा सके।

संदेश. [RFC 2136] और [RFC 3007] DNS डायनेमिक अपडेट निर्दिष्ट करते हैं.)

एक बार ये सभी चरण पूरे हो जाने के बाद, लोग आपकी वेबसाइट पर जा सकेंगे और आपकी कंपनी के कर्मचारियों को ईमेल भेज सकेंगे। आइए DNS पर अपनी चर्चा को यह सत्यापित करके समाप्त करें कि यह कथन सत्य है। यह सत्यापन DNS के बारे में हमने जो सीखा है उसे पुख्ता करने में भी मदद करता है। मान लीजिए ऑस्ट्रेलिया में ऐलिस

वेब पेज www.networkutopia.com **देखना चाहता है** जैसा कि पहले चर्चा की गई थी, उसका मेजबान सबसे पहले भेजेगा
अपने स्थानीय DNS सर्वर को DNS क्वेरी भेजें। स्थानीय DNS सर्वर फिर एक TLD com सर्वर से संपर्क करेगा।
यदि TLD com सर्वर का पता उपलब्ध नहीं है **तो स्थानीय DNS सर्वर को भी रूट DNS सर्वर से संपर्क करना होगा।**
कैश किया गया।) इस TLD सर्वर में ऊपर सूचीबद्ध प्रकार NS और प्रकार A संसाधन रिकॉर्ड शामिल हैं, क्योंकि रजिस्ट्रार ने इन संसाधन रिकॉर्ड को सभी TLD कॉम सर्वर में डाला था। TLD कॉम सर्वर एलिस के स्थानीय DNS सर्वर को एक उत्तर भेजता है, जिसमें उत्तर में दो संसाधन रिकॉर्ड शामिल होते हैं। स्थानीय

DNS सर्वर फिर 212.212.212.1 पर DNS क्वेरी भेजता है , जिसमें संबंधित टाइप A रिकॉर्ड के लिए पूछा जाता है
www.networkutopia.com पर यह रिकॉर्ड वांछित वेब सर्वर का आईपी पता प्रदान करता है, जैसे,
212.212.71.4 , जिसे स्थानीय DNS सर्वर एलिस के होस्ट को वापस भेज देता है। एलिस का ब्राउज़र अब

होस्ट 212.212.71.4 **से एक TCP कनेक्शन आरंभ करें** और कनेक्शन पर एक HTTP अनुरोध भेजें।

वाह! जब कोई वेब पर सर्फिंग करता है तो जो कुछ दिखता है, उससे कहीं अधिक चल रहा होता है!

2.5 पीयर-टू-पीयर फ़ाइल वितरण

इस अध्याय में अब तक वर्णित अनुप्रयोग - जिसमें वेब, ई-मेल और DNS शामिल हैं - सभी क्लाइंट-सर्वर आर्किटेक्चर का उपयोग करते हैं, जो हमेशा चालू रहने वाले इंफ्रास्ट्रक्चर सर्वर पर महत्वपूर्ण निर्भरता रखते हैं।

धारा 2.1.1 के अनुसार, P2P आर्किटेक्चर के साथ, हमेशा चालू रहने वाले इंफ्रास्ट्रक्चर सर्वर पर न्यूनतम (या कोई) निर्भरता नहीं होती है। इसके बजाय, बीच-बीच में जुड़े होस्ट के जोड़े, जिन्हें पीयर कहा जाता है, एक दूसरे से सीधे संवाद करते हैं। पीयर किसी सेवा प्रदाता के स्वामित्व में नहीं होते हैं, बल्कि वे डेस्कटॉप और लैपटॉप होते हैं जिन्हें उपयोगकर्ता नियंत्रित करते हैं।

इस खंड में हम एक बहुत ही स्वाभाविक P2P अनुप्रयोग पर विचार करते हैं, अर्थात्, एक बड़ी फ़ाइल को एक सर्वर से बड़ी संख्या में होस्ट (पीयर कहा जाता है) में वितरित करना। फ़ाइल लिनक्स ऑपरेटिंग सिस्टम का एक नया संस्करण, किसी मौजूदा ऑपरेटिंग सिस्टम या अनुप्रयोग के लिए एक सॉफ़्टवेयर पैच, एक MP3 संगीत फ़ाइल या एक MPEG वीडियो फ़ाइल हो सकती है। क्लाइंट-सर्वर फ़ाइल वितरण में, सर्वर को प्रत्येक पीयर को फ़ाइल की एक प्रति भेजनी चाहिए - जिससे सर्वर पर बहुत अधिक बोझ पड़ता है और सर्वर बैंडविड्थ की बड़ी मात्रा का उपभोग होता है।

पी2पी फ़ाइल वितरण में, प्रत्येक पीयर अपने द्वारा प्राप्त फ़ाइल के किसी भी हिस्से को किसी अन्य पीयर को पुनः वितरित कर सकता है, जिससे वितरण प्रक्रिया में सर्वर की सहायता होती है। 2016 तक, सबसे लोकप्रिय पी2पी फ़ाइल वितरण प्रोटोकॉल बिटटोरेंट है। मूल रूप से ब्रैम कोहेन द्वारा विकसित, अब बिटटोरेंट प्रोटोकॉल के अनुरूप कई अलग-अलग स्वतंत्र बिटटोरेंट क्लाइंट हैं, ठीक उसी तरह जैसे कई वेब ब्राउज़र क्लाइंट हैं जो HTTP प्रोटोकॉल के अनुरूप हैं। इस उपखंड में, हम सबसे पहले फ़ाइल वितरण के संदर्भ में पी2पी आर्किटेक्चर की स्व-मापनीयता की जांच करते हैं। फिर हम बिटटोरेंट का कुछ विस्तार से वर्णन करते हैं, इसकी सबसे महत्वपूर्ण विशेषताओं और विशेषताओं पर प्रकाश डालते हैं।

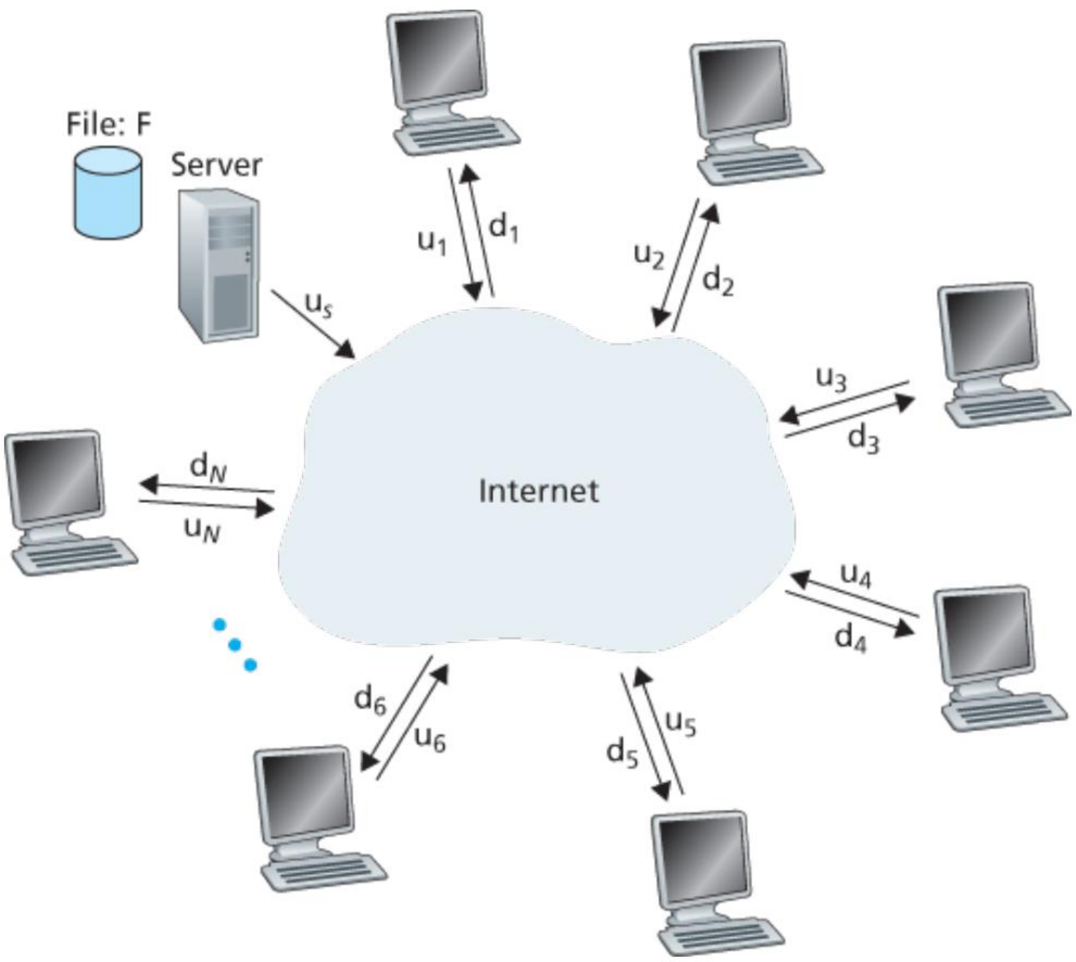
पी2पी आर्किटेक्चर की मापनीयता

क्लाइंट-सर्वर आर्किटेक्चर की तुलना पीयर-टू-पीयर आर्किटेक्चर से करने के लिए, तथा पी2पी की अंतर्निहित स्व-मापनीयता को स्पष्ट करने के लिए, अब हम एक फ़ाइल को एक निश्चित समूह में वितरित करने के लिए एक सरल मात्रात्मक मॉडल पर विचार करते हैं।

दोनों आर्किटेक्चर प्रकारों के लिए पीयर। जैसा कि चित्र 2.22 में दिखाया गया है, सर्वर और पीयर एक्सेस लिंक के साथ इंटरनेट से जुड़े हुए हैं। सर्वर के एक्सेस लिंक की अपलोड दर को u **द्वारा निरूपित करें** एस , अपलोड दर

ith पीयर का एक्सेस लिंक u से , **और** ith पीयर के एक्सेस लिंक की डाउनलोड दर d से। इसे भी निरूपित करें मैं

वितरित की जाने वाली फ़ाइल का आकार (बिट्स में) F **से** और फ़ाइल की प्रतिलिपि प्राप्त करने के इच्छुक साथियों की संख्या N **से**। वितरण समय वह समय है जो फ़ाइल प्राप्त करने में लगता है



चित्र 2.22 एक उदाहरणात्मक फ़ाइल वितरण समस्या

सभी N **साथियों** को फ़ाइल की एक प्रति । नीचे वितरण समय के हमारे विश्लेषण में, क्लाइंट-सर्वर और P2P आर्किटेक्चर दोनों के लिए, हम सरलीकृत (और आम तौर पर सटीक [अकेला 2003]) धारणा बनाते हैं कि इंटरनेट कोर में प्रचुर बैंडविड्थ है, जिसका अर्थ है कि सभी अड़चनें एक्सेस नेटवर्क में हैं। हम यह भी मानते हैं कि सर्वर और क्लाइंट किसी अन्य नेटवर्क एप्लिकेशन में भाग नहीं ले रहे हैं, ताकि उनके सभी अपलोड और डाउनलोड एक्सेस बैंडविड्थ इस फ़ाइल को वितरित करने के लिए पूरी तरह से समर्पित हो सकें।

आइए सबसे पहले क्लाइंट-सर्वर आर्किटेक्चर के लिए वितरण समय निर्धारित करें, जिसे हम D **से दर्शाते हैं** । क्लाइंट-सर्वर आर्किटेक्चर में, कोई भी सहकर्मी फ़ाइल वितरित करने में सहायता नहीं करता है। हम निम्नलिखित बनाते हैं
अवलोकन:

- सर्वर को फ़ाइल की एक कॉपी N पीयर में से प्रत्येक को भेजनी होगी। इस प्रकार सर्वर को NF बिट्स भेजनी होंगी। चूँकि सर्वर की अपलोड दर बहुत अधिक **है** एस फ़ाइल वितरित करने का समय कम से कम NF/u **होना चाहिए .** एस
- मान लीजिए d सबसे कम डाउनलोड दर वाले साथी की डाउनलोड दर को दर्शाता है, अर्थात dmin=min{d1,dp,. . .,dN}. सबसे कम डाउनलोड दर वाला पीयर फ़ाइल के **सभी** F **बिट्स प्राप्त नहीं कर सकता है** F/d **सेकंड** से कम । इस प्रकार न्यूनतम वितरण समय कम से कम F/d **है** । मिन

इन दोनों प्रेक्षणों को एक साथ रखने पर, हमें यह प्राप्त होता है

डीसीएस□अधिकतम{एनएफस,एफडीमिन}.

यह क्लाइंट-सर्वर आर्किटेक्चर के लिए न्यूनतम वितरण समय पर एक निचली सीमा प्रदान करता है। होमवर्क समस्याओं में आपको यह दिखाने के लिए कहा जाएगा कि सर्वर अपने प्रसारण को इस तरह से शेड्यूल कर सकता है कि निचली सीमा वास्तव में प्राप्त हो जाए। तो चलिए ऊपर दी गई इस निचली सीमा को वास्तविक वितरण के रूप में लेते हैं

समय, यानि,

डीसीएस=अधिकतम{एनएफयूएस,एफडीमिन} (2.1)

समीकरण 2.1 से हम देखते हैं कि N के पर्याप्त बड़े होने पर, क्लाइंट-सर्वर वितरण समय NF/u **द्वारा दिया जाता है** ।

इस प्रकार, वितरण समय साथियों की संख्या N के साथ रैखिक रूप से बढ़ता है। इसलिए, उदाहरण के लिए, यदि एक सप्ताह से अगले सप्ताह तक साथियों की संख्या एक हजार से दस लाख तक हजार गुना बढ़ जाती है, तो सभी साथियों को फ़ाइल वितरित करने के लिए आवश्यक समय 1,000 से बढ़ जाता है।

आइए अब P2P आर्किटेक्चर के लिए एक समान विश्लेषण करें, जहां प्रत्येक पीयर फ़ाइल वितरित करने में सर्वर की सहायता कर सकता है। विशेष रूप से, जब कोई पीयर कुछ फ़ाइल डेटा प्राप्त करता है, तो वह डेटा को अन्य पीयर को पुनः वितरित करने के लिए अपनी स्वयं की अपलोड क्षमता का उपयोग कर सकता है। P2P आर्किटेक्चर के लिए वितरण समय की गणना क्लाइंट-सर्वर आर्किटेक्चर की तुलना में कुछ अधिक जटिल है, क्योंकि वितरण समय इस बात पर निर्भर करता है कि प्रत्येक पीयर फ़ाइल के हिस्सों को अन्य पीयर को कैसे वितरित करता है। फिर भी, एक सरल

न्यूनतम वितरण समय के लिए अभिव्यक्ति प्राप्त की जा सकती है [कुमार 2006]। इसके लिए, हम पहले निम्नलिखित अवलोकन करते हैं:

- वितरण की शुरुआत में, केवल सर्वर के पास ही फ़ाइल होती है। इस फ़ाइल को साथियों के समुदाय में लाने के लिए, सर्वर को फ़ाइल के प्रत्येक बिट को कम से कम एक बार अपने एक्सेस लिंक में भेजना होगा। इस प्रकार, न्यूनतम

  वितरण समय कम से कम F/u **है** । (क्लाइंट-सर्वर योजना के विपरीत, सर्वर द्वारा एक बार भेजा गया बिट सर्वर को दोबारा भेजने की आवश्यकता नहीं होती है, क्योंकि सहकर्मी आपस में बिट को पुनः वितरित कर सकते हैं।)
- क्लाइंट-सर्वर आर्किटेक्चर की तरह, सबसे कम डाउनलोड दर वाला पीयर सभी F बिट्स **प्राप्त नहीं कर सकता है**

  फ़ाइल का वितरण **समय** F/d सेकंड से भी कम है। इस प्रकार न्यूनतम वितरण समय कम से कम F/d **है** । मिन
- अंत में, ध्यान दें कि पूरे सिस्टम की कुल अपलोड क्षमता सर्वर की अपलोड दर और प्रत्येक व्यक्तिगत पीयर की अपलोड दर के बराबर है, अर्थात, utotal=us+u1++uN.

  सिस्टम को प्रत्येक N **पीयर को** F **बिट्स** डिलीवर (अपलोड) करना होगा , इस प्रकार कुल NF बिट्स डिलीवर होंगे।

  u से अधिक तेज गति से नहीं किया जा सकता । इस प्रकार, न्यूनतम वितरण समय भी कम से कम है

  एनएफ/(us+u1++uN).

इन तीनों प्रेक्षणों को एक साथ रखकर, हम P2P के लिए न्यूनतम वितरण समय प्राप्त करते हैं, जिसे द्वारा दर्शाया जाता है

डी पी2पी

DP2Pmax{Fus,Fdmin,NFus+i=1Nui} (2.2)

समीकरण 2.2 पी2पी आर्किटेक्चर के लिए न्यूनतम वितरण समय के लिए एक निचली सीमा प्रदान करता है। यह पता चलता है कि अगर हम कल्पना करते हैं कि प्रत्येक सहकर्मी बिट प्राप्त करते ही बिट को पुनर्वितरित कर सकता है, तो एक है

पुनर्वितरण योजना जो वास्तव में इस निचली सीमा को प्राप्त करती है [कुमार 2006]। (हम होमवर्क में इस परिणाम का एक विशेष मामला साबित करेंगे।) वास्तव में, जहाँ फ़ाइल के हिस्सों को पुनर्वितरित किया जाता है बजाय इसके कि

व्यक्तिगत बिट्स के लिए, समीकरण 2.2 वास्तविक न्यूनतम वितरण समय का एक अच्छा अनुमान है।

इस प्रकार, आइए समीकरण 2.2 **द्वारा प्रदान की गई निचली सीमा को** वास्तविक न्यूनतम वितरण समय के रूप में लें, अर्थात,

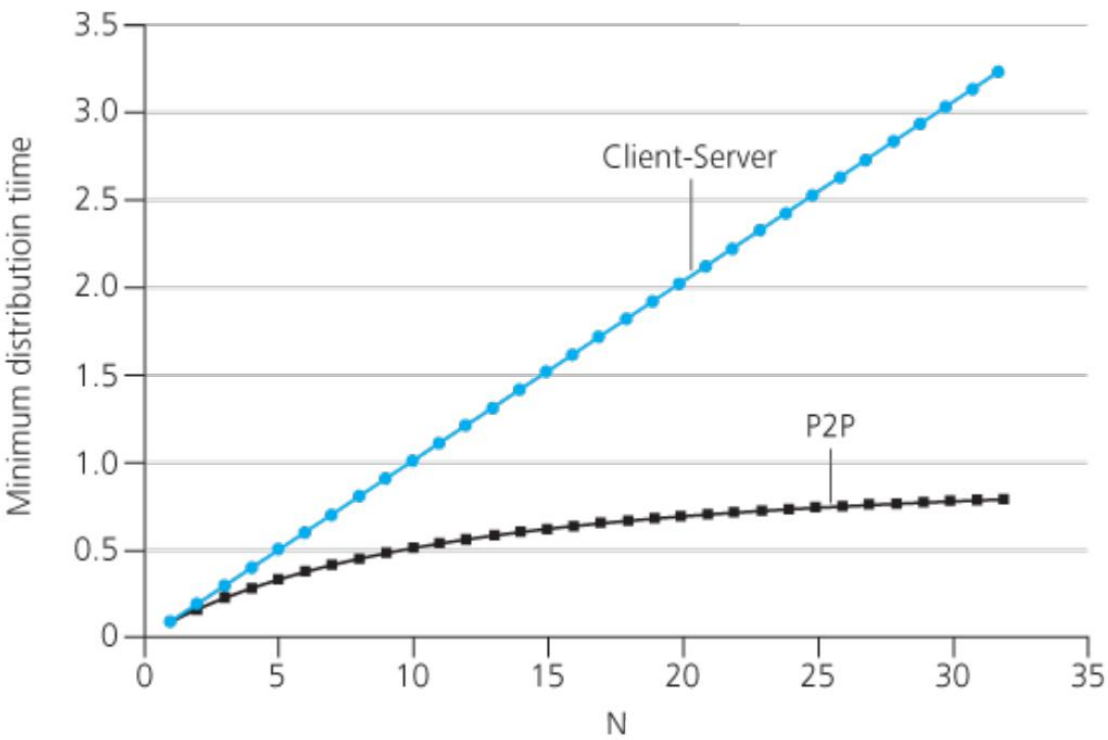
DP2P=अधिकतम{Fus,Fdmin,NFus+i=1Nui} (2.3)

चित्र 2.23 क्लाइंट-सर्वर और पी2पी आर्किटेक्चर के लिए न्यूनतम वितरण समय की तुलना करता है

कि सभी साथियों की अपलोड दर समान u है। चित्र 2.23 में, हमने dminus सेट किया है। F/u=1 घंटा, us=10u, यह मानते हुए

इस प्रकार, एक सहकर्मी एक घंटे में पूरी फ़ाइल संचारित कर सकता है, सर्वर संचरण दर 10 है

सहकर्मी अपलोड दर का गुना,



चित्र 2.23 पी2पी और क्लाइंट-सर्वर आर्किटेक्चर के लिए वितरण समय

और (सरलता के लिए) सहकर्मी डाउनलोड दरें इतनी बड़ी रखी जाती हैं कि उनका कोई प्रभाव न पड़े। हम देखते हैं

चित्र 2.23 से पता चलता है कि क्लाइंट-सर्वर आर्किटेक्चर के लिए, वितरण समय रैखिक रूप से और बिना किसी सीमा के बढ़ता है क्योंकि साथियों की संख्या बढ़ती है। हालाँकि, P2P आर्किटेक्चर के लिए, न्यूनतम वितरण समय न केवल हमेशा क्लाइंट-सर्वर आर्किटेक्चर के वितरण समय से कम होता है; यह

किसी भी संख्या में साथियों के लिए एक घंटे से भी कम समय N. इस प्रकार, P2P आर्किटेक्चर वाले एप्लिकेशन स्व-स्केलिंग हो सकते हैं। यह स्केलेबिलिटी साथियों के पुनर्वितरक होने के साथ-साथ उपभोक्ता होने का प्रत्यक्ष परिणाम है

बिट्स का.

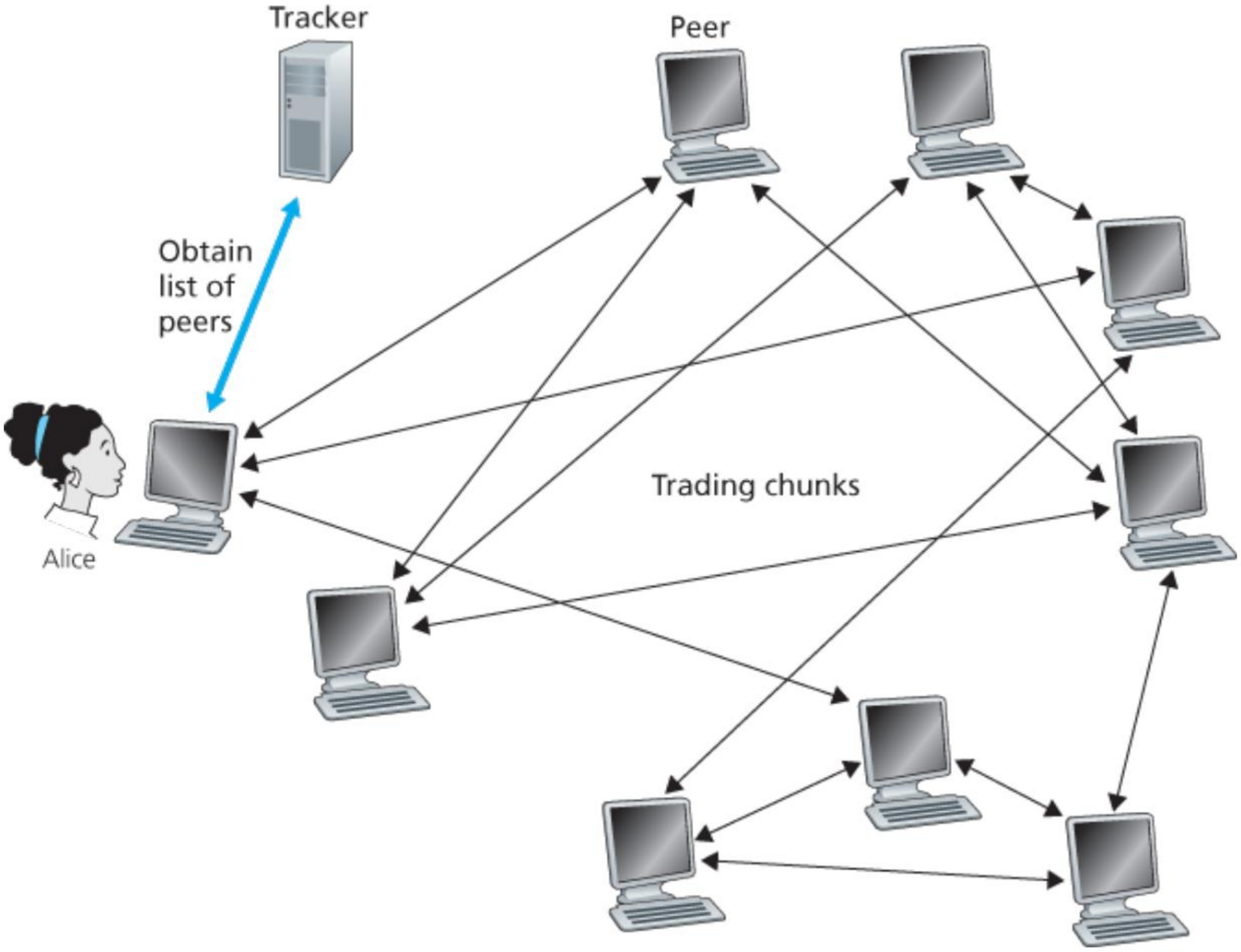
बिटटोरेंट

बिटटोरेंट फ़ाइल वितरण के लिए एक लोकप्रिय पी2पी प्रोटोकॉल है [चाओ 2011]। बिटटोरेंट भाषा में, का संग्रह

किसी विशेष फ़ाइल के वितरण में भाग लेने वाले सभी साथियों को टोरेंट कहा जाता है। टोरेंट में साथी एक दूसरे से फ़ाइल के बराबर आकार **के हिस्से डाउनलोड करते हैं ,** जिसका सामान्य हिस्सा 256 KBytes होता है। जब कोई साथी पहली बार टोरेंट से जुड़ता है, तो उसके पास कोई हिस्सा नहीं होता। समय के साथ-साथ यह अधिक से अधिक हिस्से जमा करता जाता है। जब यह हिस्सा डाउनलोड करता है तो यह अन्य साथियों को हिस्सा अपलोड भी करता है।
एक बार जब कोई साथी पूरी फ़ाइल प्राप्त कर लेता है, तो वह (स्वार्थी रूप से) टोरेंट छोड़ सकता है, या (परोपकारात्मक रूप से) टोरेंट में रह सकता है और अन्य साथियों को हिस्सा अपलोड करना जारी रख सकता है। साथ ही, कोई भी साथी किसी भी समय केवल हिस्सों के एक उपसमूह के साथ टोरेंट छोड़ सकता है, और बाद में फिर से जुड़ सकता है
धार.

आइए अब बिटटोरेंट के संचालन के बारे में विस्तार से जानें। चूंकि बिटटोरेंट एक जटिल प्रोटोकॉल और सिस्टम है, इसलिए हम केवल इसके सबसे महत्वपूर्ण तंत्रों का वर्णन करेंगे, कुछ विवरणों को छिपाएंगे; इससे हमें पेड़ों के बीच जंगल देखने का मौका मिलेगा। प्रत्येक टोरेंट का एक बुनियादी ढांचा होता है

नोड को ट्रैकर कहा जाता है।



चित्र 2.24 बिटटोरेंट के साथ फ़ाइल वितरण

जब कोई सहकर्मी टोरेंट से जुड़ता है, तो वह ट्रैकर के साथ खुद को पंजीकृत करता है और समय-समय पर ट्रैकर को सूचित करता है कि वह अभी भी टोरेंट में है। इस तरह, ट्रैकर टोरेंट में भाग लेने वाले सहकर्मियों पर नज़र रखता है।
किसी भी टोरेंट में किसी भी समय दस से कम या एक हजार से अधिक पीयर भाग ले सकते हैं।
समय।

जैसा कि चित्र 2.24 में दिखाया गया है , जब एक नया पीयर, एलिस, टोरेंट से जुड़ता है, तो ट्रैकर भाग लेने वाले पीयर के सेट से पीयर के एक उपसमूह (ठोसता के लिए, मान लें कि 50) को यादृच्छिक रूप से चुनता है, और इन 50 पीयर के आईपी पते एलिस को भेजता है। पीयर की इस सूची को प्राप्त करके, एलिस इस सूची में सभी पीयर के साथ समवर्ती TCP कनेक्शन स्थापित करने का प्रयास करती है। आइए उन सभी पीयर को कॉल करें जिनके साथ एलिस

एक TCP कनेक्शन "पड़ोसी साथियों" को स्थापित करने में सफल होता है। (चित्र 2.24 में, ऐलिस के पास केवल तीन पड़ोसी साथी दिखाए गए हैं। आम तौर पर, उसके पास कई और भी होंगे।) जैसे-जैसे समय बीतता है, इनमें से कुछ साथी छोड़ सकते हैं और अन्य साथी (शुरुआती 50 के बाहर) ऐलिस के साथ TCP कनेक्शन स्थापित करने का प्रयास कर सकते हैं। इसलिए एक साथी के पड़ोसी साथी समय के साथ उतार-चढ़ाव करेंगे।

किसी भी समय, प्रत्येक पीयर के पास फ़ाइल से कुछ भाग होंगे, जिसमें अलग-अलग पीयर के पास अलग-अलग उपसमूह होंगे। समय-समय पर, ऐलिस अपने प्रत्येक पड़ोसी पीयर से (TCP कनेक्शन पर) पूछेगी

उनके पास मौजूद चंक्स की सूची के लिए। अगर ऐलिस के पास L **अलग-अलग पड़ोसी हैं, तो उसे** चंक्स की L **सूचियाँ** मिलेंगी ।
इस जानकारी के साथ, ऐलिस उन खंडों के लिए अनुरोध जारी करेगी (पुनः TCP कनेक्शनों पर) जिन्हें वह वर्तमान में उपयोग कर रही है।
नहीं है.

इसलिए किसी भी समय, ऐलिस के पास चंक्स का एक उपसमूह होगा और उसे पता होगा कि उसके पड़ोसियों के पास कौन से चंक्स हैं। इस जानकारी के साथ, ऐलिस को दो महत्वपूर्ण निर्णय लेने होंगे। पहला, उसे अपने पड़ोसियों से सबसे पहले कौन से चंक्स का अनुरोध करना चाहिए? और दूसरा, उसे अपने पड़ोसियों में से किसको अनुरोधित चंक्स भेजना चाहिए? यह तय करने में कि कौन से चंक्स का अनुरोध करना है, ऐलिस एक तकनीक का उपयोग करती है जिसे रेयरेस्ट फर्स्ट कहा जाता है।

विचार यह है कि उसके पास जो खंड नहीं हैं, उनमें से वह खंड निर्धारित किया जाए जो उसके पड़ोसियों में सबसे दुर्लभ है (यानी, वह खंड जिसकी उसके पड़ोसियों में सबसे कम दोहराई गई प्रतियाँ हैं) और फिर उन दुर्लभतम खंडों का पहले अनुरोध करें। इस तरीके से, सबसे दुर्लभ खंड अधिक तेज़ी से पुनर्वितरित हो जाते हैं, जिसका उद्देश्य टोरेंट में प्रत्येक खंड की प्रतियों की संख्या को (मोटे तौर पर) बराबर करना है।

यह निर्धारित करने के लिए कि वह किन अनुरोधों का जवाब देती है, बिटटोरेंट एक चतुर ट्रेडिंग एल्गोरिदम का उपयोग करता है।

ऐलिस उन पड़ोसियों को प्राथमिकता देती है जो वर्तमान में उसे उच्चतम दर पर डेटा प्रदान कर रहे हैं।
विशेष रूप से, अपने प्रत्येक पड़ोसी के लिए, ऐलिस लगातार उस दर को मापती है जिस पर वह बिट्स प्राप्त करती है और उन चार साथियों को निर्धारित करती है जो उसे सबसे अधिक दर पर बिट्स खिला रहे हैं। फिर वह उन्हीं चार साथियों को चंक भेजकर प्रतिक्रिया करती है। हर 10 सेकंड में, वह दरों की पुनर्गणना करती है और संभवतः चार साथियों के सेट को संशोधित करती है। बिटटोरेंट भाषा में, इन चार साथियों को अनचोक्ड कहा जाता है।

महत्वपूर्ण बात यह है कि हर 30 सेकंड में वह यादृच्छिक रूप से एक अतिरिक्त पड़ोसी को चुनती है और उसे टुकड़े भेजती है।
चलिए यादृच्छिक रूप से चुने गए साथी को बॉब कहते हैं। बिटटोरेंट भाषा में बॉब को आशावादी रूप से अप्रभावित कहा जाता है।
क्योंकि ऐलिस बॉब को डेटा भेज रही है, वह बॉब के शीर्ष चार अपलोडर में से एक बन सकती है, जिस स्थिति में बॉब ऐलिस को डेटा भेजना शुरू कर देगा। यदि बॉब द्वारा ऐलिस को डेटा भेजने की दर काफी अधिक है, तो बॉब, बदले में, ऐलिस के शीर्ष चार अपलोडर में से एक बन सकता है। दूसरे शब्दों में, हर 30 सेकंड में, ऐलिस बेतरतीब ढंग से एक नया ट्रेडिंग पार्टनर चुनेगी और उस पार्टनर के साथ ट्रेडिंग शुरू करेगी। यदि दो सहकर्मी ट्रेडिंग से संतुष्ट हैं, तो वे एक-दूसरे को अपनी शीर्ष चार सूचियों में डाल देंगे और एक-दूसरे के साथ तब तक ट्रेडिंग जारी रखेंगे जब तक कि सहकर्मियों में से कोई एक बेहतर पार्टनर नहीं ढूंढ लेता। इसका प्रभाव यह होता है कि संगत दरों पर अपलोड करने में सक्षम सहकर्मी एक-दूसरे को ढूंढ लेते हैं। यादृच्छिक पड़ोसी चयन नए सहकर्मियों को चंक प्राप्त करने की भी अनुमति देता है, ताकि उनके पास ट्रेड करने के लिए कुछ हो। इन पाँच सहकर्मियों के अलावा सभी अन्य पड़ोसी सहकर्मी

(चार "शीर्ष" सहकर्मी और एक जांच करने वाला सहकर्मी) "घुट गए" हैं, अर्थात, उन्हें ऐलिस से कोई भी हिस्सा प्राप्त नहीं होता है।
बिटटोरेंट में कई दिलचस्प तंत्र हैं जिनकी चर्चा यहां नहीं की गई है, जिनमें टुकड़े (मिनी-चंक्स), पाइपलाइनिंग, यादृच्छिक पहला चयन, एंडगेम मोड और एंटी-स्नबिंग शामिल हैं [कोहेन 2003]।

अभी वर्णित व्यापार के लिए प्रोत्साहन तंत्र को अक्सर टिट-फॉर-टेट [कोहेन 2003] के रूप में संदर्भित किया जाता है।
दिखाया गया है कि इस प्रोत्साहन योजना को दरकिनार किया जा सकता है [लियोगकास 2006; लोचर 2006; पियाटेक 2007]। फिर भी, बिटटोरेंट पारिस्थितिकी तंत्र बेहद सफल है, जिसमें लाखों सहकर्मी एक साथ सैकड़ों हज़ारों टोरेंट में फ़ाइलों को सक्रिय रूप से साझा करते हैं। अगर बिटटोरेंट को बिना किसी प्रतिशोध (या किसी अन्य प्रकार) के डिज़ाइन किया गया होता, लेकिन अन्यथा बिल्कुल वैसा ही होता, तो बिटटोरेंट शायद अब अस्तित्व में भी नहीं होता, क्योंकि

अधिकांश उपयोगकर्ता फ्रीराइडर्स रहे होंगे [सरोइउ 2002]।

हम P2P पर अपनी चर्चा को P2P के दूसरे अनुप्रयोग, अर्थात् वितरित हैश टेबल (DHT) का संक्षिप्त उल्लेख करके समाप्त करते हैं। एक वितरित हैश टेबल एक सरल डेटाबेस है, जिसमें डेटाबेस रिकॉर्ड P2P सिस्टम में साथियों पर वितरित किए जाते हैं। DHT को व्यापक रूप से लागू किया गया है (उदाहरण के लिए, बिटटोरेंट में) और व्यापक शोध का विषय रहा है। एक अवलोकन वीडियो नोट में प्रदान किया गया है

साथी वेबसाइट.

VideoNote

वितरित हैश तालिकाओं पर चलते हुए

2.6 वीडियो स्ट्रीमिंग और सामग्री वितरण नेटवर्क

उत्तरी अमेरिका में आवासीय ISP में अब अधिकांश ट्रैफ़िक के लिए स्ट्रीमिंग प्रीरिकॉर्डेड वीडियो का उपयोग किया जाता है। विशेष रूप से, नेटफ्लिक्स और यूट्यूब सेवाओं ने अकेले 37% और 16% ट्रैफ़िक का उपभोग किया।

2015 में आवासीय ISP ट्रैफ़िक का क्रमशः [सैंडवाइन 2015]। इस अनुभाग में हम आज के इंटरनेट में लोकप्रिय वीडियो स्ट्रीमिंग सेवाओं को कैसे लागू किया जाता है, इसका अवलोकन प्रदान करेंगे। हम देखेंगे कि उन्हें एप्लिकेशन-स्तरीय प्रोटोकॉल और सर्वर का उपयोग करके लागू किया जाता है जो कुछ मायनों में कैश की तरह काम करते हैं।

मल्टीमीडिया नेटवर्किंग को समर्पित अध्याय 9 में , हम इंटरनेट वीडियो के साथ-साथ अन्य इंटरनेट मल्टीमीडिया सेवाओं की भी आगे जांच करेंगे।

2.6.1 इंटरनेट वीडियो

स्ट्रीमिंग स्टोर किए गए वीडियो एप्लीकेशन में, अंतर्निहित माध्यम पहले से रिकॉर्ड किया गया वीडियो होता है, जैसे कि कोई मूवी, कोई टेलीविज़न शो, कोई पहले से रिकॉर्ड किया गया खेल आयोजन, या कोई पहले से रिकॉर्ड किया गया उपयोगकर्ता द्वारा बनाया गया वीडियो (जैसे कि YouTube पर आमतौर पर देखा जाता है)। ये पहले से रिकॉर्ड किए गए वीडियो सर्वर पर रखे जाते हैं, और उपयोगकर्ता उन्हें भेजते हैं।

सर्वर से वीडियो को ऑन डिमांड देखने का अनुरोध किया जाता है। आज कई इंटरनेट कंपनियाँ स्ट्रीमिंग वीडियो उपलब्ध कराती हैं, जिनमें नेटफ्लिक्स, यूट्यूब (गूगल), अमेज़न और यूकू शामिल हैं।

लेकिन वीडियो स्ट्रीमिंग की चर्चा शुरू करने से पहले, हमें पहले वीडियो माध्यम के बारे में थोड़ा-बहुत जान लेना चाहिए। वीडियो छवियों का एक क्रम है, जो आम तौर पर एक स्थिर दर पर प्रदर्शित होता है, उदाहरण के लिए, प्रति सेकंड 24 या 30 छवियां। एक असम्पीडित, डिजिटल रूप से एनकोडेड छवि में पिक्सेल की एक सरणी होती है, जिसमें प्रत्येक पिक्सेल को चमक और रंग को दर्शाने के लिए कई बिट्स में एनकोड किया जाता है। वीडियो की एक महत्वपूर्ण विशेषता यह है कि इसे संपीड़ित किया जा सकता है, जिससे वीडियो की गुणवत्ता बिट दर के साथ बदल जाती है। आज के ऑफ-द-शेल्फ संपीड़न एल्गोरिदम किसी भी वांछित बिट दर पर वीडियो को संपीड़ित कर सकते हैं। बेशक, बिट दर जितनी अधिक होगी, छवि की गुणवत्ता उतनी ही बेहतर होगी और समग्र उपयोगकर्ता देखने का अनुभव उतना ही बेहतर होगा।

नेटवर्किंग के नजरिए से, शायद वीडियो की सबसे प्रमुख विशेषता इसकी उच्च बिट दर है।

संपीड़ित इंटरनेट वीडियो आमतौर पर कम गुणवत्ता वाले वीडियो के लिए 100 केबीपीएस से लेकर उच्च परिभाषा वाली फिल्मों के लिए 3 एमबीपीएस से अधिक तक होता है; 4K स्ट्रीमिंग में 10 एमबीपीएस से अधिक की बिटरेट की आवश्यकता होती है। यह विशेष रूप से उच्च-स्तरीय वीडियो के लिए बहुत अधिक ट्रैफ़िक और संग्रहण में बदल सकता है। उदाहरण के लिए, 67 मिनट की अवधि वाला एक 2 एमबीपीएस वीडियो 1 गीगाबाइट संग्रहण और ट्रैफ़िक का उपभोग करेगा। अब तक, स्ट्रीमिंग वीडियो के लिए सबसे महत्वपूर्ण प्रदर्शन माप औसत एंड-टू-एंड थ्रूपुट है। निरंतर प्लेआउट प्रदान करने के लिए, नेटवर्क को स्ट्रीमिंग एप्लिकेशन को एक औसत थ्रूपुट प्रदान करना चाहिए जो संपीड़ित वीडियो की बिट दर जितना बड़ा हो।

हम एक ही वीडियो के कई संस्करण बनाने के लिए भी संपीड़न का उपयोग कर सकते हैं, प्रत्येक संस्करण अलग-अलग गुणवत्ता स्तर पर। उदाहरण के लिए, हम एक ही वीडियो के तीन संस्करण बनाने के लिए संपीड़न का उपयोग कर सकते हैं, 300 केबीपीएस, 1 एमबीपीएस और 3 एमबीपीएस की दर से। उपयोगकर्ता तब तय कर सकते हैं कि वे अपने वर्तमान उपलब्ध बैंडविड्थ के आधार पर कौन सा संस्करण देखना चाहते हैं। हाई-स्पीड इंटरनेट कनेक्शन वाले उपयोगकर्ता 3 एमबीपीएस संस्करण चुन सकते हैं; स्मार्टफोन के साथ 3 जी पर वीडियो देखने वाले उपयोगकर्ता 300 केबीपीएस संस्करण चुन सकते हैं।

2.6.2 HTTP स्ट्रीमिंग और DASH

HTTP स्ट्रीमिंग में, वीडियो को HTTP सर्वर पर एक विशिष्ट URL के साथ एक साधारण फ़ाइल के रूप में संग्रहीत किया जाता है। जब कोई उपयोगकर्ता वीडियो देखना चाहता है, तो क्लाइंट सर्वर के साथ एक टीसीपी कनेक्शन स्थापित करता है और जारी करता है

उस URL के लिए HTTP GET अनुरोध। फिर सर्वर, अंतर्निहित नेटवर्क प्रोटोकॉल और ट्रैफ़िक स्थितियों के अनुसार, HTTP प्रतिक्रिया संदेश के भीतर वीडियो फ़ाइल को तेज़ी से भेजता है। क्लाइंट साइड पर, बाइट्स को क्लाइंट एप्लिकेशन बफ़र में एकत्र किया जाता है। एक बार जब इस बफ़र में बाइट्स की संख्या एक पूर्व निर्धारित सीमा से अधिक हो जाती है, तो क्लाइंट एप्लिकेशन प्लेबैक शुरू कर देता है - विशेष रूप से, स्ट्रीमिंग वीडियो एप्लिकेशन समय-समय पर क्लाइंट एप्लिकेशन बफ़र से वीडियो फ़्रेम को पकड़ता है, फ़्रेम को डीकंप्रेस करता है, और उन्हें उपयोगकर्ता की स्क्रीन पर प्रदर्शित करता है। इस प्रकार, वीडियो स्ट्रीमिंग एप्लिकेशन वीडियो प्रदर्शित कर रहा है क्योंकि यह वीडियो के बाद के हिस्सों के अनुरूप फ़्रेम प्राप्त और बफ़र कर रहा है।

यद्यपि HTTP स्ट्रीमिंग, जैसा कि पिछले पैराग्राफ में वर्णित है, व्यवहार में बड़े पैमाने पर तैनात की गई है (उदाहरण के लिए, YouTube द्वारा इसकी शुरुआत से), इसमें एक बड़ी कमी है: सभी क्लाइंट को वीडियो की एक ही एन्कोडिंग प्राप्त होती है, भले ही क्लाइंट के लिए उपलब्ध बैंडविड्थ की मात्रा में बड़े अंतर हों, अलग-अलग क्लाइंट के लिए और एक ही क्लाइंट के लिए समय के साथ भी। इसने HTTP-आधारित स्ट्रीमिंग के एक नए प्रकार के विकास को जन्म दिया है, जिसे अक्सर HTTP (DASH) पर डायनेमिक अडैप्टिव स्ट्रीमिंग के रूप में संदर्भित किया जाता है। DASH में, वीडियो को कई अलग-अलग संस्करणों में एन्कोड किया जाता है, प्रत्येक संस्करण में एक अलग बिट दर होती है और तदनुसार, एक अलग गुणवत्ता स्तर होता है। क्लाइंट गतिशील रूप से कुछ सेकंड की लंबाई के वीडियो सेगमेंट के टुकड़ों का अनुरोध करता है। जब उपलब्ध बैंडविड्थ की मात्रा अधिक होती है, तो क्लाइंट स्वाभाविक रूप से उच्च-दर वाले संस्करण से टुकड़ों का चयन करता है; और जब उपलब्ध बैंडविड्थ कम होती है, तो यह स्वाभाविक रूप से कम-दर वाले संस्करण से टुकड़ों का चयन करता है। क्लाइंट HTTP GET अनुरोध के साथ एक बार में एक अलग-अलग टुकड़ों का चयन करता है

संदेश [अख़शाबी 2011].

DASH विभिन्न इंटरनेट एक्सेस दरों वाले ग्राहकों को अलग-अलग एन्कोडिंग दरों पर वीडियो स्ट्रीम करने की अनुमति देता है। कम गति वाले 3G कनेक्शन वाले क्लाइंट कम बिट-रेट (और कम गुणवत्ता वाला) संस्करण प्राप्त कर सकते हैं, और फाइबर कनेक्शन वाले क्लाइंट उच्च गुणवत्ता वाला संस्करण प्राप्त कर सकते हैं। DASH क्लाइंट को उपलब्ध बैंडविड्थ के अनुकूल होने की भी अनुमति देता है यदि सत्र के दौरान उपलब्ध एंड-टू-एंड बैंडविड्थ में बदलाव होता है। यह सुविधा विशेष रूप से मोबाइल उपयोगकर्ताओं के लिए महत्वपूर्ण है, जो आमतौर पर बेस स्टेशनों के संबंध में चलते समय अपनी बैंडविड्थ उपलब्धता में उतार-चढ़ाव देखते हैं।

DASH के साथ, प्रत्येक वीडियो संस्करण HTTP सर्वर में संग्रहीत किया जाता है, प्रत्येक का एक अलग URL होता है।

सर्वर में एक मैनिफ़ेस्ट फ़ाइल भी होती है, जो प्रत्येक संस्करण के लिए एक URL और उसकी बिट दर प्रदान करती है। क्लाइंट सबसे पहले मैनिफ़ेस्ट फ़ाइल का अनुरोध करता है और विभिन्न संस्करणों के बारे में सीखता है। क्लाइंट फिर प्रत्येक चंक के लिए HTTP GET अनुरोध संदेश में एक URL और एक बाइट रेंज निर्दिष्ट करके एक बार में एक चंक का चयन करता है।

खंड डाउनलोड करते समय, क्लाइंट प्राप्त बैंडविड्थ को भी मापता है और एक दर चलाता है

अगले अनुरोध के लिए खंड का चयन करने के लिए निर्धारण एल्गोरिथ्म। स्वाभाविक रूप से, यदि क्लाइंट के पास बहुत अधिक वीडियो बफर है और यदि मापा गया प्राप्त बैंडविड्थ उच्च है, तो यह उच्च-बिटरेट संस्करण से एक खंड का चयन करेगा। और स्वाभाविक रूप से यदि क्लाइंट के पास बहुत कम वीडियो बफर है और मापा गया प्राप्त बैंडविड्थ कम है, तो यह कम-बिटरेट संस्करण से एक खंड का चयन करेगा। इसलिए DASH क्लाइंट को विभिन्न गुणवत्ता स्तरों के बीच स्वतंत्र रूप से स्विच करने की अनुमति देता है।

2.6.3 सामग्री वितरण नेटवर्क

आज, कई इंटरनेट वीडियो कंपनियाँ प्रतिदिन लाखों उपयोगकर्ताओं को ऑन-डिमांड मल्टी-एमबीपीएस स्ट्रीम वितरित कर रही हैं। उदाहरण के लिए, YouTube, करोड़ों वीडियो की लाइब्रेरी के साथ, हर दिन दुनिया भर के उपयोगकर्ताओं को करोड़ों वीडियो स्ट्रीम वितरित करता है। इस सारे ट्रैफ़िक को स्ट्रीम करना

दुनिया भर के स्थानों पर निरंतर प्लेआउट और उच्च अन्तरक्रियाशीलता प्रदान करना स्पष्ट रूप से एक है
चुनौतीपूर्ण कार्य.

इंटरनेट वीडियो कंपनी के लिए, स्ट्रीमिंग वीडियो सेवा प्रदान करने का शायद सबसे सीधा तरीका एक विशाल डेटा सेंटर बनाना, अपने सभी वीडियो को डेटा सेंटर में संग्रहीत करना और डेटा सेंटर से सीधे दुनिया भर के क्लाइंट को वीडियो स्ट्रीम करना है। लेकिन इस दृष्टिकोण के साथ तीन बड़ी समस्याएं हैं। सबसे पहले, यदि क्लाइंट डेटा सेंटर से दूर है, तो सर्वर-टू-क्लाइंट पैकेट कई संचार लिंक को पार करेंगे और संभवतः कई ISP से होकर गुजरेंगे, जिनमें से कुछ ISP संभवतः अलग-अलग महाद्वीपों पर स्थित होंगे। यदि इनमें से कोई एक लिंक वीडियो खपत दर से कम थ्रूपुट प्रदान करता है, तो एंड-टू-एंड थ्रूपुट भी खपत दर से कम होगा, जिसके परिणामस्वरूप कष्टप्रद फ्रीजिंग होगी

उपयोगकर्ता के लिए देरी। (अध्याय 1 **से याद करें** कि एक स्ट्रीम का अंत-से-अंत थ्रूपुट बोतलनेक लिंक पर थ्रूपुट द्वारा नियंत्रित होता है।) लिंक की संख्या बढ़ने के साथ ऐसा होने की संभावना बढ़ जाती है

अंत-से-अंत पथ में वृद्धि होती है। दूसरी कमी यह है कि एक लोकप्रिय वीडियो को संभवतः कई बार भेजा जाएगा

एक ही संचार लिंक पर कई बार। इससे न केवल नेटवर्क बैंडविड्थ बर्बाद होता है, बल्कि इंटरनेट भी

वीडियो कंपनी खुद अपने प्रदाता आईएसपी (डेटा सेंटर से जुड़ी) को इंटरनेट पर बार-बार एक ही **बाइट्स भेजने के लिए भुगतान करेगी। इस समाधान के साथ एक तीसरी समस्या यह है कि एक ही डेटा सेंटर**

विफलता के एकल बिंदु का प्रतिनिधित्व करता है - यदि डेटा केंद्र या इंटरनेट से उसका लिंक बंद हो जाता है, तो यह विफल नहीं होगा

किसी भी वीडियो स्ट्रीम को वितरित करने में सक्षम हो ।

दुनिया भर में फैले उपयोगकर्ताओं को भारी मात्रा में वीडियो डेटा वितरित करने की चुनौती को पूरा करने के लिए, लगभग सभी प्रमुख वीडियो-स्ट्रीमिंग कंपनियाँ कंटेंट डिस्ट्रीब्यूशन नेटवर्क (CDN) का उपयोग करती हैं। एक CDN कई भौगोलिक रूप से वितरित स्थानों में सर्वरों का प्रबंधन करता है, अपने सर्वर में वीडियो (और अन्य प्रकार की वेब सामग्री, जिसमें दस्तावेज़, चित्र और ऑडियो शामिल हैं) की प्रतियाँ संग्रहीत करता है, और प्रत्येक उपयोगकर्ता अनुरोध को एक CDN स्थान पर निर्देशित करने का प्रयास करता है जो सर्वोत्तम उपयोगकर्ता अनुभव प्रदान करेगा।

CDN एक निजी CDN हो सकता है, अर्थात, सामग्री प्रदाता के स्वामित्व में हो; उदाहरण के लिए, Google का CDN YouTube वीडियो और अन्य प्रकार की सामग्री वितरित करता है। CDN वैकल्पिक रूप से एक तृतीय-पक्ष CDN **हो सकता है** जो कई सामग्री प्रदाताओं की ओर से सामग्री वितरित करता है; अकामाई, लाइमलाइट और लेवल-3 सभी

तृतीय-पक्ष CDN संचालित करें। आधुनिक CDN का एक बहुत ही पठनीय अवलोकन [लीटन 2009; न्यग्रेन 2010] है।

सीडीएन आमतौर पर दो अलग-अलग सर्वर प्लेसमेंट दर्शन में से एक को अपनाते हैं [हुआंग 2008]:

- गहराई तक प्रवेश करें। अकामाई द्वारा अग्रणी एक दर्शन, दुनिया भर में एक्सेस आईएसपी में सर्वर क्लस्टर तैनात करके, इंटरनेट सेवा प्रदाताओं के एक्सेस नेटवर्क में गहराई से प्रवेश करना **है। (एक्सेस**

  नेटवर्क का वर्णन अनुभाग 1.3 में किया गया है।) अकामाई लगभग 1,700 स्थानों पर क्लस्टर के साथ इस दृष्टिकोण को अपनाता है। इसका लक्ष्य अंतिम उपयोगकर्ताओं के करीब पहुंचना है, जिससे उपयोगकर्ता द्वारा अनुभव की जाने वाली देरी और

  अंतिम उपयोगकर्ता और CDN सर्वर के बीच लिंक और राउटर की संख्या कम करके थ्रूपुट को कम किया जा सकता है, जिससे वह सामग्री प्राप्त करता है। इस अत्यधिक वितरित डिज़ाइन के कारण, रखरखाव और

  क्लस्टरों का प्रबंधन चुनौतीपूर्ण हो जाता है।
- घर पर लाना। लाइमलाइट और कई अन्य CDN कंपनियों द्वारा अपनाया गया दूसरा डिजाइन दर्शन, कम संख्या (उदाहरण के लिए, दसियों) साइटों पर बड़े क्लस्टर बनाकर ISP को घर पर लाना **है।**

  ये CDN आमतौर पर ISP के अंदर प्रवेश करने के बजाय अपने क्लस्टरों को इंटरनेट में रखते हैं

  एक्सचेंज पॉइंट्स (आईएक्सपी) (अनुभाग 1.3 देखें)। एंटर-डीप डिज़ाइन दर्शन की तुलना में, ब्रिंग-होम डिज़ाइन के परिणामस्वरूप आमतौर पर कम रखरखाव और प्रबंधन ओवरहेड होता है, संभवतः कम से कम

  अंतिम उपयोगकर्ताओं के लिए अधिक विलंब और कम प्रवाह की लागत।

एक बार जब इसके क्लस्टर स्थापित हो जाते हैं, तो CDN अपने क्लस्टर में सामग्री की प्रतिलिपि बनाता है। CDN प्रत्येक क्लस्टर में प्रत्येक वीडियो की एक प्रति नहीं रखना चाहेगा, क्योंकि कुछ वीडियो शायद ही कभी देखे जाते हैं या केवल कुछ देशों में लोकप्रिय होते हैं। वास्तव में, कई CDN अपने क्लस्टर में वीडियो नहीं भेजते हैं, बल्कि इसके बजाय एक सरल पुल रणनीति का उपयोग करते हैं: यदि कोई क्लाइंट किसी ऐसे क्लस्टर से वीडियो का अनुरोध करता है जो वीडियो संग्रहीत नहीं कर रहा है, तो क्लस्टर वीडियो को पुनः प्राप्त करता है (केंद्रीय रिपॉजिटरी से या किसी अन्य क्लस्टर से) और स्ट्रीमिंग के दौरान स्थानीय रूप से एक प्रति संग्रहीत करता है

वीडियो को उसी समय क्लाइंट को भेज देता है। इसी तरह की वेब कैशिंग (अनुभाग 2.2.5 देखें), जब क्लस्टर का स्टोरेज भर जाता है, तो यह उन वीडियो को हटा देता है जिन्हें अक्सर अनुरोध नहीं किया जाता है।

सीडीएन ऑपरेशन

CDN को तैनात करने के दो प्रमुख तरीकों की पहचान करने के बाद, आइए अब CDN के संचालन के बारे में विस्तार से जानें। जब उपयोगकर्ता के ब्राउज़र में कोई ब्राउज़र होता है, तो वह CDN को चलाने के लिए एक ब्राउज़र का उपयोग करता है।

> केस स्टडी
>
> गूगल का नेटवर्क इंफ्रास्ट्रक्चर
>
> अपनी क्लाउड सेवाओं की विशाल श्रृंखला का समर्थन करने के लिए—जिसमें सर्च, जीमेल, कैलेंडर, यूट्यूब वीडियो, मैप्स, दस्तावेज़ और सोशल नेटवर्क शामिल हैं—Google ने एक व्यापक निजी नेटवर्क और CDN अवसंरचना तैनात की है। Google के CDN अवसंरचना में सर्वर क्लस्टर के तीन स्तर हैं:

- चौदह "मेगा डेटा सेंटर", जिनमें से आठ उत्तरी अमेरिका में, चार यूरोप में और दो एशिया में हैं [Google स्थान 2016], प्रत्येक डेटा सेंटर में लगभग 100,000 सर्वर हैं।
  ये मेगा डेटा सेंटर गतिशील (और अक्सर वैयक्तिकृत) सामग्री प्रदान करने के लिए जिम्मेदार हैं, जिसमें खोज परिणाम और जीमेल संदेश शामिल हैं।

- दुनिया भर में फैले IXP में अनुमानित 50 क्लस्टर हैं, जिनमें से प्रत्येक क्लस्टर में 100-500 सर्वर होते हैं [अधिकारी 2011a]। ये क्लस्टर YouTube वीडियो सहित स्थिर सामग्री परोसने के लिए ज़िम्मेदार हैं [अधिकारी 2011a]।

- एक एक्सेस ISP के भीतर कई सैकड़ों "एंटर-डीप" क्लस्टर स्थित हैं। यहाँ एक क्लस्टर में आम तौर पर एक ही रैक के भीतर दसियों सर्वर होते हैं। ये एंटर-डीप सर्वर प्रदर्शन करते हैं
  टीसीपी विभाजन (अनुभाग 3.7 देखें) और स्थैतिक सामग्री प्रदान करना [चेन 2011], जिसमें खोज परिणामों को शामिल करने वाले वेब पृष्ठों के स्थैतिक भाग शामिल हैं।

ये सभी डेटा सेंटर और क्लस्टर स्थान Google के अपने निजी नेटवर्क के साथ नेटवर्क किए गए हैं। जब कोई उपयोगकर्ता कोई खोज क्वेरी करता है, तो अक्सर क्वेरी को पहले स्थानीय ISP पर पास के एंटर-डीप कैश में भेजा जाता है, जहाँ से स्थिर सामग्री प्राप्त की जाती है; क्लाइंट को स्थिर सामग्री प्रदान करते समय, पास का कैश Google के निजी नेटवर्क पर क्वेरी को मेगा डेटा सेंटर में से एक पर भी अग्रेषित करता है, जहाँ से व्यक्तिगत खोज परिणाम प्राप्त किए जाते हैं। YouTube वीडियो के लिए, वीडियो स्वयं ब्रिंग-होम कैश में से किसी एक से आ सकता है, जबकि वीडियो के आस-पास के वेब पेज के हिस्से पास के एंटर-डीप कैश से आ सकते हैं, और वीडियो के आस-पास के विज्ञापन डेटा सेंटर से आते हैं। संक्षेप में, स्थानीय ISP को छोड़कर, Google क्लाउड सेवाएँ बड़े पैमाने पर एक नेटवर्क अवसंरचना द्वारा प्रदान की जाती हैं जो सार्वजनिक इंटरनेट से स्वतंत्र है।

होस्ट को एक विशिष्ट वीडियो (URL द्वारा पहचाना गया) प्राप्त करने का निर्देश दिया जाता है, CDN को अनुरोध को रोकना चाहिए ताकि वह (1) उस समय उस क्लाइंट के लिए एक उपयुक्त CDN सर्वर क्लस्टर निर्धारित कर सके, और (2) क्लाइंट के अनुरोध को उस क्लस्टर में एक सर्वर पर पुनर्निर्देशित कर सके। हम जल्द ही चर्चा करेंगे कि CDN एक उपयुक्त क्लस्टर कैसे निर्धारित कर सकता है। लेकिन पहले आइए एक अनुरोध को रोकने और पुनर्निर्देशित करने के पीछे के तंत्र की जाँच करें।

अधिकांश CDN अनुरोधों को रोकने और पुनर्निर्देशित करने के लिए DNS का लाभ उठाते हैं; इस तरह की एक दिलचस्प चर्चा

DNS का उपयोग [Vixie 2009] है। आइए एक सरल उदाहरण पर विचार करें कि DNS आमतौर पर कैसे शामिल होता है। मान लीजिए कि एक कंटेंट प्रदाता, NetCinema, अपने ग्राहकों को अपने वीडियो वितरित करने के लिए तीसरे पक्ष की CDN कंपनी, KingCDN को नियुक्त करता है। NetCinema वेब पेजों पर, इसके प्रत्येक वीडियो को एक URL दिया जाता है जिसमें स्ट्रिंग "वीडियो" और वीडियो के लिए एक अद्वितीय पहचानकर्ता शामिल होता है; उदाहरण के लिए, ट्रांसफॉर्मर

7 को http://video.netcinema.com/6Y7B23V असाइन किया जा सकता है। इसके बाद छह चरण होते हैं, जैसा कि चित्र में दिखाया गया है 2.25:

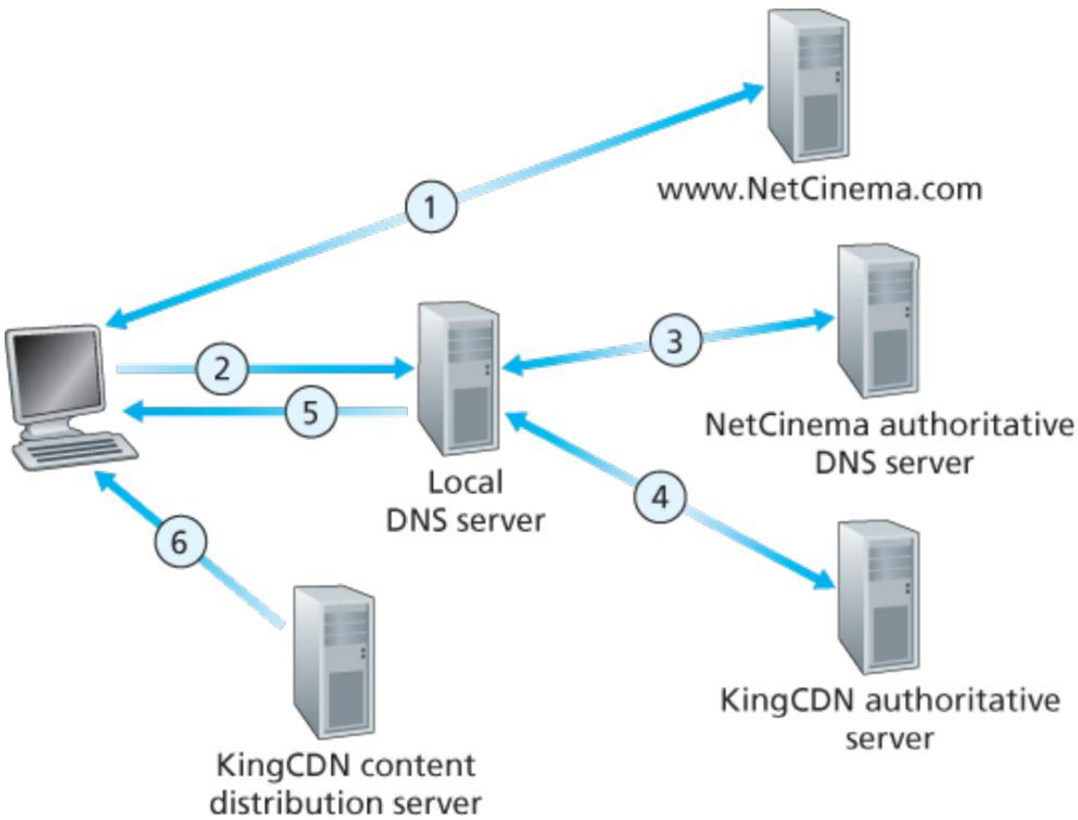
1. उपयोगकर्ता नेटसिनेमा पर वेब पेज पर जाता है।
2. जब उपयोगकर्ता http://video.netcinema.com/6Y7B23V लिंक पर क्लिक करता है, उपयोगकर्ता का होस्ट एक संदेश भेजता है video.netcinema.com के लिए DNS क्वेरी .

3. उपयोगकर्ता का स्थानीय DNS सर्वर (LDNS) DNS क्वेरी को एक आधिकारिक DNS सर्वर पर रिले करता है

नेटसिनेमा, जो होस्टनाम video.netcinema.com में स्ट्रिंग "वीडियो" देखता है । DNS क्वेरी को KingCDN को "सौंपने" के लिए, IP पता लौटाने के बजाय, NetCinema आधिकारिक DNS सर्वर LDNS को KingCDN के डोमेन में एक होस्टनाम लौटाता है,

उदाहरण, a1105.kingcdn.com.

4. इस बिंदु से, DNS क्वेरी KingCDN के निजी DNS इंफ्रास्ट्रक्चर में प्रवेश करती है। उपयोगकर्ता का LDNS फिर एक दूसरी क्वेरी भेजता है, अब a1105.kingcdn.com के लिए, और KingCDN का DNS सिस्टम अंततः LDNS को KingCDN सामग्री सर्वर के IP पते लौटाता है। इस प्रकार, KingCDN के DNS सिस्टम के भीतर, वह CDN सर्वर निर्दिष्ट किया जाता है जिससे क्लाइंट अपनी सामग्री प्राप्त करेगा।



चित्र 2.25 DNS उपयोगकर्ता के अनुरोध को CDN सर्वर पर पुनर्निर्देशित करता है

5. LDNS सामग्री-सेवारत CDN नोड का IP पता उपयोगकर्ता के होस्ट को अग्रेषित करता है।

6. जब क्लाइंट को KingCDN कंटेंट सर्वर के लिए IP पता प्राप्त होता है, तो वह उस IP पते पर सर्वर के साथ एक सीधा TCP कनेक्शन स्थापित करता है और वीडियो के लिए HTTP GET अनुरोध जारी करता है। यदि DASH का उपयोग किया जाता है, तो सर्वर पहले क्लाइंट को URL की एक सूची के साथ एक मैनिफ़ेस्ट फ़ाइल भेजेगा, वीडियो के प्रत्येक संस्करण के लिए एक, और क्लाइंट गतिशील रूप से विभिन्न संस्करणों से विखंडन का चयन करेगा।

संस्करण.

क्लस्टर चयन रणनीतियाँ

किसी भी CDN परिनियोजन के मूल में एक क्लस्टर चयन रणनीति होती है, यानी क्लाइंट को CDN के भीतर सर्वर क्लस्टर या डेटा सेंटर पर गतिशील रूप से निर्देशित करने का एक तंत्र। जैसा कि हमने अभी देखा,

CDN क्लाइंट के DNS लुकअप के माध्यम से क्लाइंट के LDNS सर्वर का IP पता सीखता है। इस IP पते को जानने के बाद, CDN को इस IP पते के आधार पर एक उपयुक्त क्लस्टर का चयन करना होगा। CDN आमतौर पर मालिकाना क्लस्टर चयन रणनीतियों को नियोजित करते हैं। अब हम कुछ दृष्टिकोणों का संक्षेप में सर्वेक्षण करते हैं, जिनमें से प्रत्येक

इसके अपने फायदे और नुकसान हैं।

एक सरल रणनीति यह है कि क्लाइंट को उस क्लस्टर को सौंपा जाए जो भौगोलिक दृष्टि से सबसे नज़दीक हो ।

वाणिज्यिक भू-स्थान डेटाबेस (जैसे कि क्वोवा [क्वोवा 2016] और मैक्स-माइंड [मैक्समाइंड 2016]) में, प्रत्येक एलडीएनएस आईपी पते को भौगोलिक स्थान पर मैप किया जाता है। जब किसी विशेष एलडीएनएस से डीएनएस अनुरोध प्राप्त होता है, तो सीडीएन भौगोलिक रूप से निकटतम क्लस्टर चुनता है, यानी वह क्लस्टर जो एलडीएनएस से सबसे कम किलोमीटर दूर है "जैसा कि पक्षी उड़ता है।" ऐसा समाधान एक बड़े के लिए काफी अच्छी तरह से काम कर सकता है

क्लाइंट का अंश [अग्रवाल 2009]। हालाँकि, कुछ क्लाइंट के लिए, समाधान खराब प्रदर्शन कर सकता है, क्योंकि भौगोलिक रूप से निकटतम क्लस्टर नेटवर्क पथ की लंबाई या हॉप्स की संख्या के संदर्भ में निकटतम क्लस्टर नहीं हो सकता है। इसके अलावा, सभी DNS-आधारित दृष्टिकोणों में निहित एक समस्या यह है कि कुछ

अंतिम उपयोगकर्ताओं को दूरस्थ रूप से स्थित LDNS का उपयोग करने के लिए कॉन्फ़िगर किया गया है [शेख 2001; माओ 2002], जिस स्थिति में LDNS स्थान क्लाइंट के स्थान से बहुत दूर हो सकता है। इसके अलावा, यह सरल रणनीति इंटरनेट पथों के समय के साथ देरी और उपलब्ध बैंडविड्थ में भिन्नता को अनदेखा करती है, हमेशा एक ही क्लस्टर को असाइन करती है

विशेष ग्राहक.

वर्तमान **ट्रैफ़िक स्थितियों** के आधार पर क्लाइंट के लिए सर्वोत्तम क्लस्टर निर्धारित करने के लिए , CDN उनके बीच विलंब और हानि प्रदर्शन का **आवधिक** वास्तविक समय माप कर सकते हैं।

क्लस्टर और क्लाइंट। उदाहरण के लिए, एक CDN अपने प्रत्येक क्लस्टर को समय-समय पर जांच (क्लस्टर और क्लाइंट के लिए) भेज सकता है।

उदाहरण के लिए, पिंग संदेश या DNS क्वेरी) को दुनिया भर के सभी LDNS पर भेजना। इसका एक नुकसान यह है

दृष्टिकोण यह है कि कई एलडीएनएस ऐसी जांचों का जवाब नहीं देने के लिए कॉन्फ़िगर किए गए हैं।

2.6.4 केस स्टडीज़: नेटफ्लिक्स, यूट्यूब और कंकन

हम तीन बेहद सफल बड़े पैमाने पर तैनाती पर नज़र डालकर संग्रहित वीडियो स्ट्रीमिंग पर अपनी चर्चा समाप्त करते हैं: नेटफ्लिक्स, यूट्यूब और कंकन। हम देखेंगे कि इनमें से प्रत्येक सिस्टम एक बहुत ही अलग दृष्टिकोण अपनाता है, फिर भी इस खंड में चर्चा किए गए कई अंतर्निहित सिद्धांतों को लागू करता है।

NetFlix

2015 में उत्तरी अमेरिका में आवासीय आईएसपी में 37% डाउनस्ट्रीम ट्रैफिक उत्पन्न करने वाले नेटफ्लिक्स ने

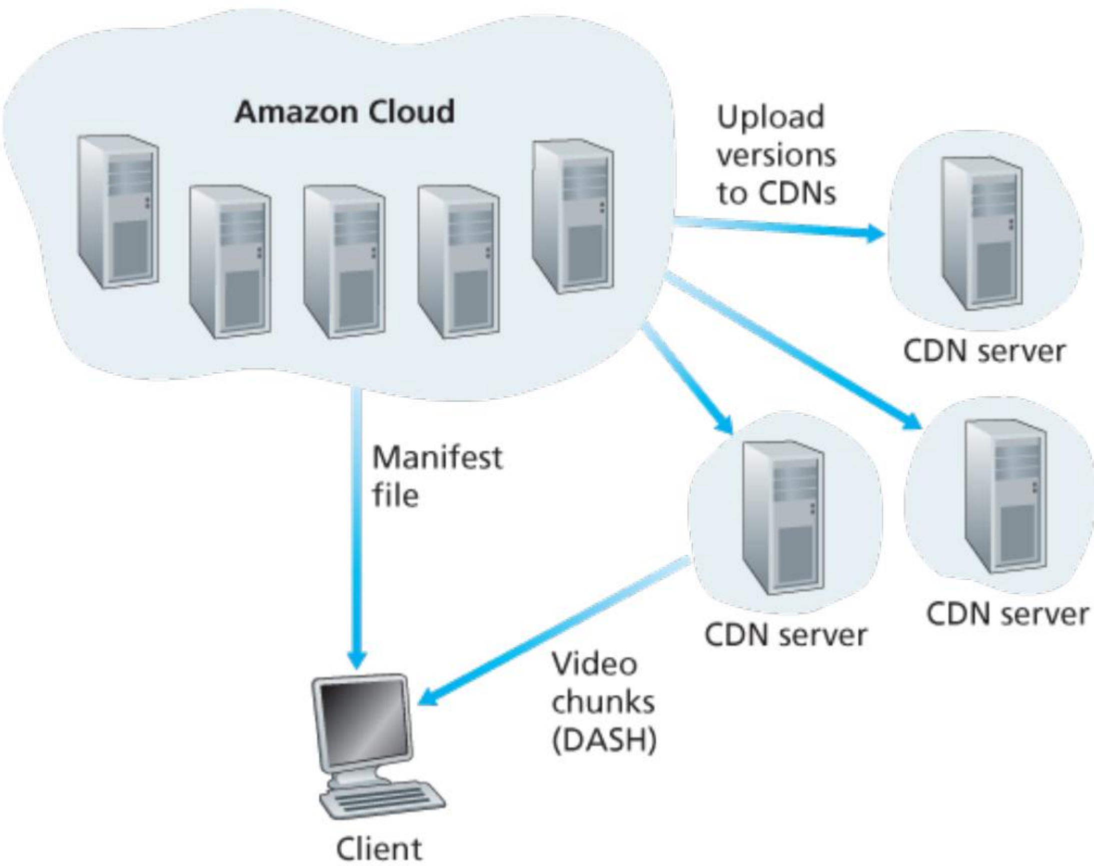
संयुक्त राज्य अमेरिका में ऑनलाइन फिल्मों और टीवी श्रृंखला के लिए अग्रणी सेवा प्रदाता बनें [सैंडवाइन 2015]। जैसा कि हम नीचे चर्चा करते हैं, नेटफ्लिक्स वीडियो वितरण के दो प्रमुख घटक हैं: अमेज़ॅन क्लाउड और इसका अपना निजी सीडीएन इंफ्रास्ट्रक्चर।

नेटफ्लिक्स की एक वेबसाइट है जो उपयोगकर्ता पंजीकरण और लॉगिन, बिलिंग सहित कई कार्यों को संभालती है।

ब्राउज़िंग और खोज के लिए मूवी कैटलॉग, और मूवी अनुशंसा प्रणाली। जैसा कि चित्र **में दिखाया गया है**

2.26, यह वेब साइट (और इससे जुड़े बैकएंड डेटाबेस) पूरी तरह से अमेज़ॅन क्लाउड में अमेज़ॅन सर्वर पर चलते हैं। इसके अतिरिक्त, अमेज़ॅन क्लाउड निम्नलिखित महत्वपूर्ण कार्यों को संभालता है:

- कंटेंट इनजेक्शन। नेटफ्लिक्स को अपने ग्राहकों को मूवी वितरित करने से पहले, उसे पहले मूवी को इनजेक्ट और प्रोसेस करना होगा। नेटफ्लिक्स मूवी के स्टूडियो मास्टर वर्जन प्राप्त करता है और उन्हें होस्ट में अपलोड करता है।
  अमेज़न क्लाउड.
- सामग्री प्रसंस्करण। अमेज़ॅन क्लाउड में मशीनें प्रत्येक मूवी के लिए कई अलग-अलग प्रारूप बनाती हैं, जो डेस्कटॉप कंप्यूटर, स्मार्टफ़ोन और टेलीविज़न से जुड़े गेम कंसोल पर चलने वाले क्लाइंट वीडियो प्लेयर की विविधता के लिए उपयुक्त हैं। इनमें से प्रत्येक प्रारूप के लिए एक अलग संस्करण बनाया जाता है और कई बिट दरों पर, जो DASH का उपयोग करके HTTP पर अनुकूली स्ट्रीमिंग की अनुमति देता है।
- अपने CDN पर संस्करण अपलोड करना। एक बार जब किसी मूवी के सभी संस्करण बना लिए जाते हैं, तो अमेज़न क्लाउड में होस्ट अपने CDN पर संस्करणों को अपलोड कर देते हैं।



चित्र 2.26 नेटफ्लिक्स वीडियो स्ट्रीमिंग प्लेटफ़ॉर्म

जब नेटफ्लिक्स ने 2007 में पहली बार अपनी वीडियो स्ट्रीमिंग सेवा शुरू की थी, तो उसने अपनी वीडियो सामग्री वितरित करने के लिए तीन तृतीय-पक्ष CDN कंपनियों को नियुक्त किया था। नेटफ्लिक्स ने तब से अपना खुद का निजी CDN बनाया है, जहाँ से वह अब अपने सभी वीडियो स्ट्रीम करता है। (हालांकि, नेटफ्लिक्स अभी भी अपने वेब पेज वितरित करने के लिए अकामाई का उपयोग करता है।)
अपने स्वयं के CDN के साथ, नेटफ्लिक्स ने IXPs और आवासीय ISPs दोनों में सर्वर रैक स्थापित किए हैं।

नेटफ्लिक्स के पास वर्तमान में 50 से अधिक IXP स्थानों पर सर्वर रैक हैं; **नेटफ्लिक्स** रैक वाले IXP की वर्तमान सूची के लिए [नेटफ्लिक्स ओपन कनेक्ट 2016] **देखें। नेटफ्लिक्स रैक वाले सैकड़ों ISP स्थान भी हैं;**

[नेटफ्लिक्स ओपन कनेक्ट 2016] भी देखें , जहाँ नेटफ्लिक्स संभावित ISP भागीदारों को उनके नेटवर्क के लिए (मुफ़्त) नेटफ्लिक्स रैक स्थापित करने के बारे में निर्देश प्रदान करता है। रैक में प्रत्येक सर्वर में कई 10 Gbps हैं

ईथरनेट पोर्ट और 100 टेराबाइट से अधिक स्टोरेज। एक रैक में सर्वरों की संख्या अलग-अलग होती है: IXP इंस्टॉलेशन में अक्सर दर्जनों सर्वर होते हैं और इसमें संपूर्ण नेटफ्लिक्स स्ट्रीमिंग वीडियो लाइब्रेरी होती है, जिसमें DASH को सपोर्ट करने वाले वीडियो के कई संस्करण शामिल होते हैं; स्थानीय IXP में केवल एक सर्वर हो सकता है और इसमें केवल एक ही सर्वर हो सकता है

सबसे लोकप्रिय वीडियो। नेटफ्लिक्स आईएक्सपी और आईएसपी में अपने सीडीएन सर्वर को पॉपुलेट करने के लिए पुल-कैशिंग (धारा 2.2.5) **का उपयोग नहीं करता** है। इसके बजाय, नेटफ्लिक्स ऑफ-पीक घंटों के दौरान अपने सीडीएन सर्वर पर वीडियो को पुश करके वितरित करता है। उन स्थानों के लिए जो पूरी लाइब्रेरी को होल्ड नहीं कर सकते, नेटफ्लिक्स केवल सबसे लोकप्रिय वीडियो पुश करता है, जो दिन-प्रतिदिन के आधार पर निर्धारित किए जाते हैं। नेटफ्लिक्स सीडीएन डिज़ाइन का कुछ विस्तार से वर्णन किया गया है

यूट्यूब वीडियो [नेटफ्लिक्स वीडियो 1] और [नेटफ्लिक्स वीडियो 2] में।

नेटफ्लिक्स आर्किटेक्चर के घटकों का वर्णन करने के बाद, आइए इंटरैक्शन पर करीब से नज़र डालें क्लाइंट और मूवी डिलीवरी में शामिल विभिन्न सर्वरों के बीच। जैसा कि पहले बताया गया है, नेटफ्लिक्स वीडियो लाइब्रेरी ब्राउज़ करने के लिए वेब पेज अमेज़ॅन क्लाउड में सर्वर से परोसे जाते हैं। जब कोई उपयोगकर्ता चलाने के लिए कोई मूवी चुनता है, तो अमेज़ॅन क्लाउड में चलने वाला नेटफ्लिक्स सॉफ़्टवेयर सबसे पहले यह निर्धारित करता है कि उसके CDN सर्वर में से किस पर मूवी की कॉपी है। मूवी वाले सर्वर में से, सॉफ़्टवेयर फिर उस क्लाइंट अनुरोध के लिए "सर्वश्रेष्ठ" सर्वर निर्धारित करता है। यदि क्लाइंट एक आवासीय ISP का उपयोग कर रहा है जिसमें उस ISP में एक नेटफ्लिक्स CDN सर्वर रैक स्थापित है, और इस रैक में अनुरोधित मूवी की एक कॉपी है, तो इस रैक में एक सर्वर आमतौर पर चुना जाता है। यदि नहीं, तो पास के IXP में एक सर्वर आमतौर पर चुना जाता है।

एक बार जब नेटफ्लिक्स उस CDN सर्वर को निर्धारित कर लेता है जो कंटेंट डिलीवर करेगा, तो वह क्लाइंट को उस विशिष्ट सर्वर का IP पता और साथ ही एक मैनिफ़ेस्ट फ़ाइल भेजता है, जिसमें अनुरोधित मूवी के विभिन्न संस्करणों के URL होते हैं। क्लाइंट और वह CDN सर्वर तब सीधे मालिकाना संस्करण का उपयोग करके इंटरैक्ट करते हैं।

डैश। विशेष रूप से, जैसा कि अनुभाग 2.6.2 में वर्णित है, क्लाइंट HTTP GET अनुरोध संदेशों में बाइट-रेंज हेडर का उपयोग करता है, मूवी के विभिन्न संस्करणों से विखंडन का अनुरोध करने के लिए। नेटफ्लिक्स उन विखंडनों का उपयोग करता है जो

लगभग चार सेकंड लंबे होते हैं [अधिकारी 2012]। जब चंक्स डाउनलोड किए जा रहे होते हैं, तो क्लाइंट प्राप्त थ्रूपुट को मापता है और दर-निर्धारण एल्गोरिदम चलाता है ताकि यह निर्धारित किया जा सके कि

अगले खंड की गुणवत्ता का अनुरोध किया जाना चाहिए।

नेटफ्लिक्स इस खंड में पहले चर्चा किए गए कई प्रमुख सिद्धांतों को शामिल करता है, जिसमें अनुकूली स्ट्रीमिंग और CDN वितरण शामिल है। हालाँकि, क्योंकि नेटफ्लिक्स अपने निजी CDN का उपयोग करता है, जो केवल वीडियो (और वेब पेज नहीं) वितरित करता है, नेटफ्लिक्स अपने CDN डिज़ाइन को सरल और अनुकूलित करने में सक्षम रहा है। विशेष रूप से,

नेटफ्लिक्स को किसी विशेष क्लाइंट को CDN सर्वर से कनेक्ट करने के लिए DNS रीडायरेक्ट का उपयोग करने की आवश्यकता नहीं है, जैसा कि अनुभाग 2.6.3 **में चर्चा की गई है** ; इसके बजाय, नेटफ्लिक्स सॉफ्टवेयर (अमेज़ॅन क्लाउड में चल रहा है) सीधे क्लाइंट को बताता है

किसी विशेष CDN सर्वर का उपयोग करने के लिए। इसके अलावा, नेटफ्लिक्स CDN पुल कैशिंग के बजाय पुश कैशिंग का उपयोग करता है

कैशिंग (धारा 2.2.5): सामग्री को कैश मिस के दौरान गतिशील रूप से नहीं, बल्कि ऑफ-पीक घंटों में निर्धारित समय पर सर्वर में डाला जाता है।

यूट्यूब

यूट्यूब पर हर मिनट 300 घंटे का वीडियो अपलोड किया जाता है और प्रतिदिन कई अरब वीडियो देखे जाते हैं

[यूट्यूब 2016], यूट्यूब निर्विवाद रूप से दुनिया की सबसे बड़ी वीडियो-शेयरिंग साइट है। यूट्यूब ने अपनी शुरुआत की

अप्रैल 2005 में सेवा शुरू की गई और नवंबर 2006 में इसे गूगल द्वारा अधिग्रहित कर लिया गया। हालाँकि गूगल/यूट्यूब का डिज़ाइन और प्रोटोकॉल स्वामित्व वाला है, फिर भी कई स्वतंत्र माप प्रयासों के माध्यम से हम एक बेहतर परिणाम प्राप्त कर सकते हैं।

YouTube कैसे काम करता है, इसके बारे में बुनियादी समझ [ज़िंक 2009; टोरेस 2011; अधिकारी 2011ए]। नेटफ्लिक्स की तरह, YouTube अपने वीडियो वितरित करने के लिए CDN तकनीक का व्यापक उपयोग करता है [टोरेस 2011]। नेटफ्लिक्स की तरह, Google YouTube वीडियो वितरित करने के लिए अपने निजी CDN का उपयोग करता है, और उसने कई सैकड़ों अलग-अलग IXP और ISP स्थानों पर सर्वर क्लस्टर स्थापित किए हैं। इन स्थानों से और सीधे अपने

विशाल डेटा केंद्रों में, Google YouTube वीडियो वितरित करता है [अधिकारी 2011a]। हालाँकि, नेटफ्लिक्स के विपरीत, Google पुल कैशिंग का उपयोग करता है, जैसा कि अनुभाग 2.2.5 में वर्णित है, और DNS रीडायरेक्ट, जैसा कि अनुभाग 2.6.3 में वर्णित है। अधिकांश समय, Google की क्लस्टर-चयन रणनीति क्लाइंट को उस क्लस्टर पर निर्देशित करती है जिसके लिए क्लाइंट और क्लस्टर के बीच RTT सबसे कम है; हालाँकि, क्लस्टर में लोड को संतुलित करने के लिए,

कभी-कभी क्लाइंट को (डीएनएस के माध्यम से) अधिक दूर के क्लस्टर पर निर्देशित किया जाता है [टोरेस 2011]।

YouTube HTTP स्ट्रीमिंग का उपयोग करता है, जो अक्सर एक वीडियो के लिए अलग-अलग संस्करणों की एक छोटी संख्या उपलब्ध कराता है, जिनमें से प्रत्येक की बिट दर और संबंधित गुणवत्ता स्तर अलग-अलग होता है। YouTube अनुकूली स्ट्रीमिंग (जैसे DASH) का उपयोग नहीं करता है, बल्कि इसके बजाय उपयोगकर्ता को मैन्युअल रूप से एक संस्करण चुनने की आवश्यकता होती है। सहेजने के लिए
बैंडविड्थ और सर्वर संसाधन जो पुनःस्थापन या समयपूर्व समाप्ति से बर्बाद हो जाएंगे, यूट्यूब
वीडियो की लक्षित मात्रा को प्रीफ़ेच करने के बाद प्रेषित डेटा के प्रवाह को सीमित करने के लिए HTTP बाइट रेंज अनुरोध का उपयोग करता है।

YouTube पर हर दिन कई मिलियन वीडियो अपलोड किए जाते हैं। न केवल YouTube वीडियो HTTP पर सर्वर से क्लाइंट तक स्ट्रीम किए जाते हैं, बल्कि YouTube अपलोडर भी HTTP पर क्लाइंट से सर्वर तक अपने वीडियो अपलोड करते हैं। YouTube अपने द्वारा प्राप्त प्रत्येक वीडियो को प्रोसेस करता है, उसे YouTube वीडियो फ़ॉर्मेट में परिवर्तित करता है और अलग-अलग बिट दरों पर कई संस्करण बनाता है। यह प्रोसेसिंग पूरी तरह से Google डेटा सेंटर के भीतर होती है।

(अनुभाग 2.6.3 में Google के नेटवर्क बुनियादी ढांचे पर केस स्टडी देखें ।)

कंकन

हमने अभी देखा कि निजी CDN द्वारा संचालित समर्पित सर्वर, क्लाइंट को Netflix और YouTube वीडियो स्ट्रीम करते हैं। Netflix और YouTube को न केवल सर्वर हार्डवेयर के लिए बल्कि सर्वर द्वारा वीडियो वितरित करने के लिए उपयोग किए जाने वाले बैंडविड्थ के लिए भी भुगतान करना पड़ता है। इन सेवाओं के पैमाने और उनके द्वारा खपत की जाने वाली बैंडविड्थ की मात्रा को देखते हुए, इस तरह की CDN तैनाती महंगी हो सकती है।

हम इस खंड का समापन बड़े पैमाने पर इंटरनेट पर मांग पर वीडियो प्रदान करने के लिए एक पूरी तरह से अलग दृष्टिकोण का वर्णन करके करते हैं - जो सेवा प्रदाता को अपने बुनियादी ढांचे और बैंडविड्थ लागत को काफी कम करने की अनुमति देता है। जैसा कि आप अनुमान लगा सकते हैं, यह दृष्टिकोण क्लाइंट-सर्वर डिलीवरी के बजाय (या उसके साथ) पी2पी डिलीवरी का उपयोग करता है। 2011 से, कंकन (ज़ुनलेई के स्वामित्व और संचालन में)

हर महीने लाखों उपयोगकर्ताओं के साथ बड़ी सफलता के साथ पी2पी वीडियो डिलीवरी की तैनाती [झांग 2015]।

उच्च स्तर पर, पी2पी वीडियो स्ट्रीमिंग बिटटोरेंट फ़ाइल डाउनलोडिंग के समान ही है। जब कोई सहकर्मी चाहता है

जब कोई वीडियो देखता है, तो यह सिस्टम में मौजूद अन्य साथियों को खोजने के लिए ट्रैकर से संपर्क करता है, जिनके पास उस वीडियो की कॉपी है। यह अनुरोध करने वाला पीयर फिर वीडियो के अन्य पीयर से समानांतर रूप से वीडियो के कुछ हिस्सों का अनुरोध करता है जिनके पास वीडियो है। हालाँकि, बिटटोरेंट के साथ डाउनलोड करने से अलग, अनुरोध प्राथमिकता से हिस्सों के लिए किए जाते हैं जिन्हें निरंतर प्लेबैक सुनिश्चित करने के लिए निकट भविष्य में फिर से चलाया जाना है [धुंगेल 2012]।

हाल ही में, कंकन ने हाइब्रिड CDN-P2P स्ट्रीमिंग सिस्टम [झांग 2015] पर माइग्रेट किया है। विशेष रूप से, कंकन अब चीन के भीतर कुछ सौ सर्वर तैनात करता है और इन सर्वरों पर वीडियो सामग्री भेजता है। यह कंकन CDN वीडियो स्ट्रीमिंग के स्टार्ट-अप चरण में एक प्रमुख भूमिका निभाता है। ज़्यादातर मामलों में, क्लाइंट CDN सर्वर से कंटेंट की शुरुआत का अनुरोध करता है, और समानांतर रूप से साथियों से कंटेंट का अनुरोध करता है। जब कुल P2P ट्रैफ़िक वीडियो प्लेबैक के लिए पर्याप्त होता है, तो क्लाइंट CDN से स्ट्रीमिंग बंद कर देगा और केवल साथियों से स्ट्रीम करेगा। लेकिन अगर P2P स्ट्रीमिंग ट्रैफ़िक अपर्याप्त हो जाता है, तो क्लाइंट CDN कनेक्शन को फिर से शुरू करेगा और हाइब्रिड CDN-P2P स्ट्रीमिंग के मोड पर वापस आ जाएगा। इस तरह, कंकन कम से कम महंगे इंफ्रास्ट्रक्चर सर्वर और बैंडविड्थ पर निर्भर रहते हुए कम शुरुआती स्टार्ट-अप देरी सुनिश्चित कर सकता है।

2.7 सॉकेट प्रोग्रामिंग: नेटवर्क अनुप्रयोग बनाना

अब जबकि हमने कई महत्वपूर्ण नेटवर्क अनुप्रयोगों पर नज़र डाल ली है, तो आइए देखें कि नेटवर्क कैसे काम करता है।

एप्लिकेशन प्रोग्राम वास्तव में बनाए जाते हैं। सेक्शन 2.1 **से याद करें** कि एक सामान्य नेटवर्क एप्लिकेशन में प्रोग्रामों की एक जोड़ी होती है - एक क्लाइंट प्रोग्राम और एक सर्वर प्रोग्राम - जो दो अलग-अलग एंड सिस्टम में रहते हैं। जब इन दो प्रोग्रामों को निष्पादित किया जाता है, तो एक क्लाइंट प्रक्रिया और एक सर्वर प्रक्रिया बनाई जाती है, और ये प्रक्रियाएँ सॉकेट से पढ़कर और लिखकर एक दूसरे के साथ संचार करती हैं। इसलिए नेटवर्क एप्लिकेशन बनाते समय, डेवलपर का मुख्य कार्य क्लाइंट और सर्वर प्रोग्राम दोनों के लिए कोड लिखना होता है।

नेटवर्क एप्लीकेशन दो प्रकार के होते हैं। एक प्रकार वह कार्यान्वयन है जिसका संचालन किसी प्रोटोकॉल मानक में निर्दिष्ट होता है, जैसे कि RFC या कोई अन्य मानक दस्तावेज़; ऐसे एप्लीकेशन को कभी-कभी "खुला" कहा जाता है, क्योंकि इसके संचालन को निर्दिष्ट करने वाले नियम सभी को ज्ञात होते हैं।

ऐसे कार्यान्वयन के लिए, क्लाइंट और सर्वर प्रोग्राम को RFC द्वारा निर्धारित नियमों के अनुरूप होना चाहिए। उदाहरण के लिए, क्लाइंट प्रोग्राम HTTP के क्लाइंट साइड का कार्यान्वयन हो सकता है

प्रोटोकॉल, जिसे सेक्शन 2.2 **में वर्णित किया गया है** और RFC 2616 में सटीक रूप से परिभाषित किया गया है; इसी तरह, सर्वर प्रोग्राम HTTP सर्वर प्रोटोकॉल का कार्यान्वयन हो सकता है, जिसे RFC 2616 में भी सटीक रूप से परिभाषित किया गया है। यदि एक डेवलपर क्लाइंट प्रोग्राम के लिए कोड लिखता है और दूसरा डेवलपर सर्वर प्रोग्राम के लिए कोड लिखता है, और दोनों डेवलपर RFC के नियमों का सावधानीपूर्वक पालन करते हैं, तो दोनों प्रोग्राम आपस में जुड़ सकेंगे। वास्तव में, आज के कई नेटवर्क अनुप्रयोगों में क्लाइंट और सर्वर प्रोग्राम के बीच संचार शामिल होता है, जिन्हें स्वतंत्र डेवलपर्स द्वारा बनाया गया है - उदाहरण के लिए, एक Google Chrome ब्राउज़र Apache वेब सर्वर के साथ संचार करता है, या एक BitTorrent क्लाइंट

बिटटोरेंट ट्रैकर.

नेटवर्क एप्लीकेशन का दूसरा प्रकार एक मालिकाना नेटवर्क एप्लीकेशन है। इस मामले में क्लाइंट और

सर्वर प्रोग्राम एक एप्लीकेशन-लेयर प्रोटोकॉल का उपयोग करते हैं जिसे **RFC या अन्यत्र खुले तौर पर प्रकाशित** नहीं किया गया है। एक एकल डेवलपर (या विकास दल) क्लाइंट और सर्वर दोनों प्रोग्राम बनाता है, और डेवलपर के पास कोड में क्या जाता है, इस पर पूरा नियंत्रण होता है। लेकिन चूँकि कोड एक खुले प्रोटोकॉल को लागू नहीं करता है, इसलिए अन्य स्वतंत्र डेवलपर ऐसा कोड विकसित नहीं कर पाएँगे जो एप्लिकेशन के साथ इंटरऑपरेट करता हो।

इस खंड में, हम क्लाइंट-सर्वर एप्लिकेशन विकसित करने में मुख्य मुद्दों की जांच करेंगे, और हम एक बहुत ही सरल क्लाइंट-सर्वर एप्लिकेशन को लागू करने वाले कोड को देखकर "अपने हाथों को गंदा करेंगे"। विकास चरण के दौरान, डेवलपर को सबसे पहले यह निर्णय लेना चाहिए कि एप्लिकेशन को TCP पर चलाना है या UDP पर। याद रखें कि TCP कनेक्शन उन्मुख है और एक विश्वसनीय बाइट-स्ट्रीम चैनल प्रदान करता है जिसके माध्यम से डेटा दो एंड सिस्टम के बीच प्रवाहित होता है। UDP कनेक्शन रहित है और डिलीवरी के बारे में किसी भी गारंटी के बिना, एक एंड सिस्टम से दूसरे तक डेटा के स्वतंत्र पैकेट भेजता है।

यह भी याद रखें कि जब कोई क्लाइंट या सर्वर प्रोग्राम RFC द्वारा परिभाषित प्रोटोकॉल को क्रियान्वित करता है, तो उसे प्रोटोकॉल से संबद्ध सुविदित पोर्ट संख्या का उपयोग करना चाहिए; इसके विपरीत, किसी स्वामित्व अनुप्रयोग को विकसित करते समय, डेवलपर को ऐसे सुविदित पोर्ट संख्याओं का उपयोग करने से बचने के लिए सावधान रहना चाहिए। (पोर्ट संख्याएँ)

अनुभाग 2.1 में संक्षेप में चर्चा की गई है । अध्याय 3 में उन पर अधिक विस्तार से चर्चा की गई है।)

हम एक सरल UDP एप्लीकेशन और एक सरल TCP एप्लीकेशन के माध्यम से UDP और TCP सॉकेट प्रोग्रामिंग का परिचय देते हैं। हम Python 3 में सरल UDP और TCP एप्लीकेशन प्रस्तुत करते हैं। हम कोड को Java, C या C++ में लिख सकते थे, लेकिन हमने Python को इसलिए चुना क्योंकि Python मुख्य सॉकेट अवधारणाओं को स्पष्ट रूप से उजागर करता है। Python के साथ कोड की कम लाइनें होती हैं, और प्रत्येक लाइन को नौसिखिए प्रोग्रामर को बिना किसी कठिनाई के समझाया जा सकता है। लेकिन अगर आप Python से परिचित नहीं हैं तो डरने की कोई ज़रूरत नहीं है।

यदि आपके पास जावा, सी या सी++ में प्रोग्रामिंग का अनुभव है तो आप आसानी से कोड का अनुसरण कर सकेंगे।

यदि आप जावा के साथ क्लाइंट-सर्वर प्रोग्रामिंग में रुचि रखते हैं, तो आपको इस पाठ्यपुस्तक के लिए कंपेनियन वेबसाइट देखने के लिए प्रोत्साहित किया जाता है; वास्तव में, आप जावा में इस अनुभाग (और संबंधित प्रयोगशालाओं) के सभी उदाहरण वहां पा सकते हैं। जो पाठक C में क्लाइंट-सर्वर प्रोग्रामिंग में रुचि रखते हैं, उनके लिए कई अच्छे उदाहरण हैं

संदर्भ उपलब्ध हैं [डोनाहू 2001; स्टीवंस 1997; फ्रॉस्ट 1994; कुरोज़ 1996]; नीचे दिए गए हमारे पायथन उदाहरणों का स्वरूप और अनुभव C के समान है।

2.7.1 UDP के साथ सॉकेट प्रोग्रामिंग

इस उप-अनुभाग में, हम सरल क्लाइंट-सर्वर प्रोग्राम लिखेंगे जो UDP का उपयोग करते हैं; अगले अनुभाग में, हम ऐसे समान प्रोग्राम लिखेंगे जो TCP का उपयोग करते हैं।

खंड 2.1 से याद करें कि विभिन्न मशीनों पर चलने वाली प्रक्रियाएँ सॉकेट में संदेश भेजकर एक दूसरे से संवाद करती हैं। हमने कहा कि प्रत्येक प्रक्रिया एक घर के समान है और प्रक्रिया का सॉकेट एक दरवाजे के समान है। एप्लिकेशन घर में दरवाजे के एक तरफ रहता है; ट्रांसपोर्ट-लेयर प्रोटोकॉल बाहरी दुनिया में दरवाजे के दूसरी तरफ रहता है। एप्लिकेशन डेवलपर के पास सॉकेट के एप्लिकेशन-लेयर साइड पर सब कुछ नियंत्रित होता है; हालाँकि, ट्रांसपोर्ट-लेयर साइड पर उसका बहुत कम नियंत्रण होता है।

अब आइए UDP सॉकेट का उपयोग करने वाली दो संचार प्रक्रियाओं के बीच की बातचीत पर करीब से नज़र डालें। इससे पहले कि भेजने की प्रक्रिया डेटा के पैकेट को सॉकेट के दरवाज़े से बाहर धकेल सके, UDP का उपयोग करते समय, उसे पहले पैकेट में एक गंतव्य पता संलग्न करना होगा। पैकेट के प्रेषक के सॉकेट से गुज़रने के बाद, इंटरनेट इस गंतव्य पते का उपयोग करके पैकेट को इंटरनेट के माध्यम से प्राप्त करने की प्रक्रिया में सॉकेट तक पहुंचाएगा। जब पैकेट प्राप्त करने वाले सॉकेट पर पहुंचता है, तो प्राप्त करने वाली प्रक्रिया सॉकेट के माध्यम से पैकेट को पुनः प्राप्त करेगी, और फिर पैकेट की सामग्री का निरीक्षण करेगी और उचित कदम उठाएगी

कार्रवाई।

अब आप सोच रहे होंगे कि पैकेट से जुड़े गंतव्य पते में क्या लिखा होता है?

जैसा कि आप उम्मीद कर सकते हैं, गंतव्य होस्ट का आईपी पता गंतव्य पते का हिस्सा है। पैकेट में गंतव्य आईपी पता शामिल करके, इंटरनेट में राउटर पैकेट को इंटरनेट के माध्यम से गंतव्य होस्ट तक रूट करने में सक्षम होंगे। लेकिन क्योंकि एक होस्ट कई नेटवर्क एप्लिकेशन प्रक्रियाएँ चला सकता है, जिनमें से प्रत्येक में एक या अधिक सॉकेट होते हैं, इसलिए विशेष पहचान करना भी आवश्यक है

गंतव्य होस्ट में सॉकेट। जब कोई सॉकेट बनाया जाता है, तो उसे एक पहचानकर्ता, जिसे पोर्ट नंबर कहा जाता है, असाइन किया जाता है। इसलिए, जैसा कि आप उम्मीद कर सकते हैं, पैकेट के गंतव्य पते में सॉकेट का पोर्ट नंबर भी शामिल है।
संक्षेप में, भेजने की प्रक्रिया पैकेट को गंतव्य पता संलग्न करती है, जिसमें गंतव्य होस्ट का आईपी पता और गंतव्य सॉकेट का पोर्ट नंबर शामिल होता है। इसके अलावा, जैसा कि हम जल्द ही देखेंगे, प्रेषक का स्रोत पता - जिसमें स्रोत होस्ट का आईपी पता और स्रोत सॉकेट का पोर्ट नंबर शामिल होता है - भी पैकेट से जुड़ा होता है। हालाँकि, पैकेट में स्रोत पता संलग्न करना

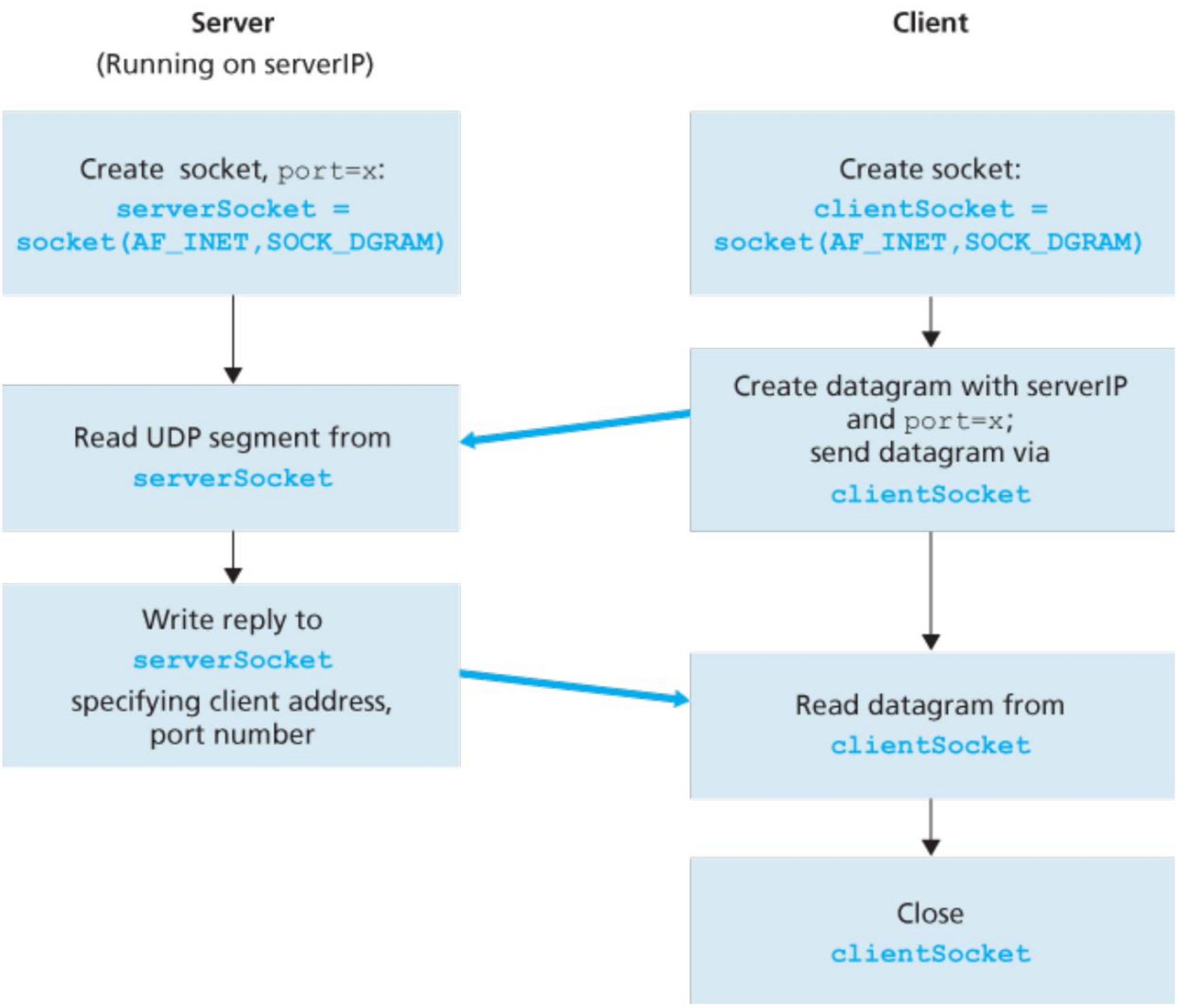
यह कार्य सामान्यतः UDP अनुप्रयोग कोड द्वारा नहीं **किया जाता है; इसके बजाय यह कार्य अंतर्निहित ऑपरेटिंग सिस्टम द्वारा स्वचालित रूप से किया जाता है।**

हम दोनों के लिए सॉकेट प्रोग्रामिंग को प्रदर्शित करने के लिए निम्नलिखित सरल क्लाइंट-सर्वर एप्लिकेशन का उपयोग करेंगे
यूडीपी और टीसीपी:

1. क्लाइंट अपने कीबोर्ड से वर्णों (डेटा) की एक पंक्ति पढ़ता है और डेटा को सर्वर पर भेजता है।
2. सर्वर डेटा प्राप्त करता है और अक्षरों को बड़े अक्षरों में परिवर्तित करता है।
3. सर्वर संशोधित डेटा क्लाइंट को भेजता है।
4. ग्राहक संशोधित डेटा प्राप्त करता है और अपनी स्क्रीन पर लाइन प्रदर्शित करता है।

चित्र 2.27, UDP ट्रांसपोर्ट सेवा पर संचार करने वाले क्लाइंट और सर्वर की मुख्य सॉकेट-संबंधी गतिविधि पर प्रकाश डालता है।

अब आइए अपने हाथों को गंदा करें और इस सरल अनुप्रयोग के UDP कार्यान्वयन के लिए क्लाइंट-सर्वर प्रोग्राम जोड़ी पर एक नज़र डालें। हम प्रत्येक प्रोग्राम के बाद एक विस्तृत, लाइन-बाय-लाइन विश्लेषण भी प्रदान करते हैं। हम UDP क्लाइंट से शुरू करेंगे, जो सर्वर को एक सरल अनुप्रयोग-स्तरीय संदेश भेजेगा।



चित्र 2.27 UDP का उपयोग करने वाला क्लाइंट-सर्वर अनुप्रयोग

क्लाइंट के संदेश को प्राप्त करने और उसका उत्तर देने में सक्षम होने के लिए सर्वर को तैयार और चालू रहना चाहिए - अर्थात, क्लाइंट द्वारा संदेश भेजे जाने से पहले इसे एक प्रक्रिया के रूप में चलना चाहिए।

क्लाइंट प्रोग्राम को UDPClient.py कहा जाता है, और सर्वर प्रोग्राम को UDPServer.py कहा जाता है। मुख्य मुद्दों पर जोर देने के लिए, हम जानबूझकर न्यूनतम कोड प्रदान करते हैं। "अच्छा कोड" में निश्चित रूप से कुछ और सहायक लाइनें होंगी, विशेष रूप से त्रुटि मामलों को संभालने के लिए। इस एप्लिकेशन के लिए, हमने सर्वर पोर्ट नंबर के लिए मनमाने ढंग से 12000 चुना है।

UDPClient.py

एप्लिकेशन के क्लाइंट साइड का कोड यहां दिया गया है:

```
सॉकेट से आयात करें *
सर्वरनाम = 'होस्टनाम'
सर्वरपोर्ट = 12000
```

```
clientSocket = सॉकेट(AF_INET, SOCK_DGRAM) संदेश = raw_input('इनपुट
लोअरकेस वाक्य:')
clientSocket.sendto(संदेश.encode(),(serverName, serverPort))
संशोधित संदेश, सर्वर पता = clientSocket.recvfrom(2048)
प्रिंट(संशोधित संदेश.डिकोड())
क्लाइंट सॉकेट.क्लोज()
```

अब आइए UDPClient.py में कोड की विभिन्न पंक्तियों पर नज़र डालें।

```
सॉकेट से आयात करें *
```

सॉकेट मॉड्यूल पायथन में सभी नेटवर्क संचार का आधार बनाता है। इस लाइन को शामिल करके, हम हमारे प्रोग्राम के भीतर सॉकेट बनाने में सक्षम होंगे।

```
सर्वरनाम = 'होस्टनाम'
सर्वरपोर्ट = 12000
```

पहली पंक्ति चर serverName को स्ट्रिंग 'hostname' पर सेट करती है। यहाँ, हम सर्वर का IP पता (जैसे, "128.138.32.126") या सर्वर का होस्टनाम (जैसे, "cis.poly.edu") युक्त स्ट्रिंग प्रदान करते हैं। यदि हम होस्टनाम का उपयोग करते हैं, तो DNS लुकअप स्वचालित रूप से प्राप्त करने के लिए किया जाएगा

आईपी पता) दूसरी पंक्ति पूर्णांक चर serverPort को 12000 पर सेट करती है।

```
क्लाइंट सॉकेट = सॉकेट (AF_INET, SOCK_DGRAM)
```

यह लाइन क्लाइंट सॉकेट बनाती है, जिसे clientSocket **कहते हैं** । पहला पैरामीटर पता बताता है

परिवार; विशेष रूप से, AF_INET इंगित करता है कि अंतर्निहित नेटवर्क IPv4 का उपयोग कर रहा है। (चिंता मत करो

अब यह - हम अध्याय 4 में IPv4 पर चर्चा करेंगे।) दूसरा पैरामीटर इंगित करता है कि सॉकेट प्रकार का है

जिसका मतलब है कि यह एक UDP सॉकेट है (TCP सॉकेट के बजाय)। ध्यान दें कि हम SOCK_DGRAM **नहीं हैं** ,

क्लाइंट सॉकेट बनाते समय उसका पोर्ट नंबर निर्दिष्ट करना; इसके बजाय हम ऑपरेटिंग सिस्टम को यह काम करने देते हैं। अब जबकि क्लाइंट प्रोसेस का डोर बन गया है, हम डोर के ज़रिए भेजने के लिए एक संदेश बनाना चाहेंगे।

```
संदेश = raw_input('इनपुट लोअरकेस वाक्य:')
```

raw_input() पायथन में एक अंतर्निहित फ़ंक्शन है। जब यह कमांड निष्पादित होता है, तो क्लाइंट पर उपयोगकर्ता को "इनपुट लोअरकेस वाक्य:" शब्दों के साथ संकेत दिया जाता है, फिर उपयोगकर्ता एक पंक्ति इनपुट करने के लिए अपने कीबोर्ड का उपयोग करता है,

जिसे message वेरिएबल में डाला जाता है । अब जब हमारे पास एक सॉकेट और एक संदेश है, तो हम चाहेंगे

सॉकेट के माध्यम से गंतव्य होस्ट तक संदेश भेजें।

```
clientSocket.sendto(संदेश.encode(),(serverName, serverPort))
```

उपरोक्त पंक्ति में, हम सबसे पहले संदेश को स्ट्रिंग प्रकार से बाइट प्रकार में परिवर्तित करते हैं, क्योंकि हमें बाइट्स भेजने की आवश्यकता होती है

सॉकेट में; यह encode() **विधि के साथ किया जाता है।** sendto() विधि गंतव्य को जोड़ती है

पता (सर्वरनाम, सर्वरपोर्ट) संदेश में जोड़ता है और परिणामी पैकेट को सर्वर में भेजता है।

प्रक्रिया का सॉकेट, clientSocket । (जैसा कि पहले बताया गया है, स्रोत पता भी पैकेट से जुड़ा होता है, हालांकि यह कोड द्वारा स्पष्ट रूप से नहीं बल्कि स्वचालित रूप से किया जाता है।) UDP सॉकेट के माध्यम से क्लाइंट-टू-सर्वर संदेश भेजना इतना सरल है! पैकेट भेजने के बाद, क्लाइंट डेटा प्राप्त करने के लिए प्रतीक्षा करता है

सर्वर.

```
संशोधित संदेश, सर्वर पता = clientSocket.recvfrom(2048)
```

उपरोक्त पंक्ति के अनुसार, जब कोई पैकेट इंटरनेट से क्लाइंट के सॉकेट पर आता है, तो पैकेट का डेटा

संशोधित संदेश चर में डाल दिया जाता है और पैकेट का स्रोत पता चर में डाल दिया जाता है

serverAddress . चर serverAddress में सर्वर का IP पता और सर्वर का IP पता दोनों शामिल होते हैं.

सर्वर का पोर्ट नंबर। UDPClient प्रोग्राम को वास्तव में इस सर्वर पते की जानकारी की आवश्यकता नहीं है, क्योंकि यह पहले से ही सर्वर पते को जानता है; लेकिन पायथन की यह पंक्ति सर्वर को पोर्ट नंबर प्रदान करती है।

फिर भी पता। विधि recvfrom भी इनपुट के रूप में बफर आकार 2048 लेता है। (यह बफर आकार अधिकांश उद्देश्यों के लिए काम करता है।)

```
प्रिंट(संशोधित संदेश.डिकोड())
```

यह लाइन उपयोगकर्ता के डिस्प्ले पर संदेश को बाइट्स से स्ट्रिंग में बदलने के बाद संशोधित संदेश प्रिंट करती है। यह वही मूल लाइन होनी चाहिए जिसे उपयोगकर्ता ने टाइप किया था, लेकिन अब कैपिटल में।

```
क्लाइंट सॉकेट.क्लोज()
```

यह लाइन सॉकेट को बंद कर देती है। फिर प्रक्रिया समाप्त हो जाती है।

UDPServer.py

आइए अब एप्लिकेशन के सर्वर साइड पर एक नज़र डालें:

```
सॉकेट से आयात करें *
सर्वरपोर्ट = 12000
सर्वर सॉकेट = सॉकेट (AF_INET, SOCK_DGRAM)
सर्वर सॉकेट.बाइंड(('', सर्वर पोर्ट))
प्रिंट("सर्वर प्राप्त करने के लिए तैयार है")
जबकि सत्य:
        संदेश, clientAddress = serverSocket.recvfrom(2048)
        संशोधितसंदेश = संदेश.डिकोड().ऊपरी()
        serverSocket.sendto(संशोधित संदेश.encode(), clientAddress)
```

ध्यान दें कि UDPServer की शुरुआत UDPClient के समान है। यह सॉकेट मॉड्यूल को भी आयात करता है,

पूर्णांक चर serverPort को 12000 पर सेट करता है, और SOCK_DGRAM (एक UDP सॉकेट) प्रकार का सॉकेट भी बनाता है । कोड की पहली पंक्ति जो UDPClient से काफी अलग है:

```
सर्वर सॉकेट.बाइंड(('', सर्वर पोर्ट))
```

उपरोक्त पंक्ति पोर्ट नंबर 12000 को सर्वर के सॉकेट से बांधती है (अर्थात असाइन करती है)। इस प्रकार UDPServer में, कोड (एप्लिकेशन डेवलपर द्वारा लिखा गया) स्पष्ट रूप से सॉकेट को पोर्ट नंबर असाइन कर रहा है। इस तरीके से, जब कोई भी सर्वर के IP पते पर पोर्ट 12000 पर पैकेट भेजता है, तो वह पैकेट इस सॉकेट पर निर्देशित किया जाएगा। UDPServer फिर एक while लूप में प्रवेश करता है; while लूप UDPServer को क्लाइंट से पैकेट प्राप्त करने और उन्हें अनिश्चित काल तक संसाधित करने की अनुमति देगा। while लूप में, UDPServer पैकेट के आने का इंतजार करता है।

```
संदेश, clientAddress = serverSocket.recvfrom(2048)
```

कोड की यह पंक्ति UDPClient में देखी गई लाइन के समान है। जब कोई पैकेट सर्वर के सॉकेट पर आता है,

पैकेट का डेटा परिवर्तनीय संदेश में डाल दिया जाता है और पैकेट का स्रोत पता परिवर्तनीय संदेश में डाल दिया जाता है।

वेरिएबल clientAddress . वेरिएबल clientAddress में क्लाइंट का IP पता और पता दोनों शामिल होते हैं.

क्लाइंट का पोर्ट नंबर। यहाँ, UDPServer इस पते की जानकारी का उपयोग करेगा , क्योंकि यह रिटर्न प्रदान करता है

पता, सामान्य डाक मेल के साथ वापसी पते के समान। इस स्रोत पते की जानकारी के साथ, अब सर्वर को पता है कि उसे अपना उत्तर कहां भेजना है।

```
संशोधितसंदेश = संदेश.डिकोड().ऊपरी()
```

यह लाइन हमारे सरल एप्लीकेशन का दिल है। यह क्लाइंट द्वारा भेजी गई लाइन लेता है और उसे कन्वर्ट करने के बाद संदेश को स्ट्रिंग में बदलने के लिए, उसे कैपिटल करने के लिए upper() **विधि का उपयोग किया जाता है।**

```
serverSocket.sendto(संशोधित संदेश.encode(), clientAddress)
```

यह अंतिम पंक्ति क्लाइंट के पते (आईपी पते और पोर्ट नंबर) को कैपिटलाइज़्ड संदेश (स्ट्रिंग को बाइट्स में बदलने के बाद) से जोड़ती है, और परिणामी पैकेट को सर्वर के सॉकेट में भेजती है। (जैसा कि पहले बताया गया है, सर्वर का पता भी पैकेट से जुड़ा होता है, हालाँकि यह कोड द्वारा स्पष्ट रूप से नहीं बल्कि स्वचालित रूप से किया जाता है।) फिर इंटरनेट इस क्लाइंट पते पर पैकेट पहुँचाएगा। सर्वर द्वारा पैकेट भेजने के बाद, यह while लूप में रहता है, दूसरे UDP पैकेट के आने का इंतज़ार करता है (किसी भी होस्ट पर चल रहे किसी भी क्लाइंट से)।

प्रोग्राम की जोड़ी का परीक्षण करने के लिए, आप एक होस्ट पर UDPClient.py और दूसरे होस्ट पर UDPServer.py चलाते हैं। UDPClient.py में सर्वर का उचित होस्टनाम या IP पता शामिल करना सुनिश्चित करें। इसके बाद, आप सर्वर होस्ट में संकलित सर्वर प्रोग्राम UDPServer.py निष्पादित करते हैं। यह सर्वर में एक प्रक्रिया बनाता है जो तब तक निष्क्रिय रहता है जब तक कि कोई क्लाइंट उससे संपर्क न कर ले। फिर आप क्लाइंट में संकलित क्लाइंट प्रोग्राम UDPClient.py निष्पादित करते हैं। यह क्लाइंट में एक प्रक्रिया बनाता है। अंत में, क्लाइंट पर एप्लिकेशन का उपयोग करने के लिए, आप एक वाक्य टाइप करते हैं जिसके बाद कैरिज रिटर्न होता है।

अपना खुद का UDP क्लाइंट-सर्वर एप्लिकेशन विकसित करने के लिए, आप क्लाइंट या सर्वर प्रोग्राम को थोड़ा संशोधित करके शुरू कर सकते हैं। उदाहरण के लिए, सभी अक्षरों को बड़े अक्षरों में बदलने के बजाय, सर्वर गिन सकता है

अक्षर s के **प्रकट होने** की संख्या और यह संख्या लौटाएँ। या आप क्लाइंट को संशोधित कर सकते हैं ताकि कैपिटलाइज़्ड वाक्य प्राप्त करने के बाद, उपयोगकर्ता सर्वर को और वाक्य भेजना जारी रख सके।

2.7.2 टीसीपी के साथ सॉकेट प्रोग्रामिंग

यूडीपी के विपरीत, टीसीपी एक कनेक्शन-उन्मुख प्रोटोकॉल है। इसका मतलब यह है कि क्लाइंट और सर्वर एक दूसरे को डेटा भेजना शुरू करने से पहले, उन्हें पहले हैंडशेक करना होगा और एक टीसीपी कनेक्शन स्थापित करना होगा। एक छोर

टीसीपी कनेक्शन का एक सिरा क्लाइंट सॉकेट से जुड़ा होता है और दूसरा सिरा सर्वर सॉकेट से जुड़ा होता है।

टीसीपी कनेक्शन बनाते समय, हम इसके साथ क्लाइंट सॉकेट एड्रेस (आईपी एड्रेस और पोर्ट) जोड़ते हैं

नंबर) और सर्वर सॉकेट पता (आईपी पता और पोर्ट नंबर)। टीसीपी कनेक्शन स्थापित होने के साथ, जब एक पक्ष दूसरे पक्ष को डेटा भेजना चाहता है, तो वह डेटा को अपने सॉकेट के माध्यम से टीसीपी कनेक्शन में छोड़ देता है। यह यूडीपी से अलग है, जिसके लिए सर्वर को पैकेट को सॉकेट में छोड़ने से पहले उसमें एक गंतव्य पता संलग्न करना चाहिए।

अब आइए TCP में क्लाइंट और सर्वर प्रोग्राम की बातचीत पर करीब से नज़र डालें। क्लाइंट का काम सर्वर से संपर्क शुरू करना है। क्लाइंट के शुरुआती संपर्क पर प्रतिक्रिया करने में सक्षम होने के लिए, सर्वर को तैयार रहना चाहिए। इसका मतलब दो बातें हैं। सबसे पहले, UDP के मामले में, क्लाइंट द्वारा संपर्क शुरू करने का प्रयास करने से पहले TCP सर्वर को एक प्रक्रिया के रूप में चलना चाहिए। दूसरा, सर्वर प्रोग्राम में एक विशेष द्वार होना चाहिए - अधिक सटीक रूप से, एक विशेष सॉकेट - जो किसी मनमाने होस्ट पर चल रहे क्लाइंट प्रक्रिया से कुछ शुरुआती संपर्क का स्वागत करता है। प्रक्रिया/सॉकेट के लिए हमारे घर/दरवाजे के सादृश्य का उपयोग करते हुए, हम कभी-कभी क्लाइंट के शुरुआती संपर्क को "स्वागत द्वार पर दस्तक" के रूप में संदर्भित करेंगे।

सर्वर प्रक्रिया चालू होने पर, क्लाइंट प्रक्रिया सर्वर से TCP कनेक्शन आरंभ कर सकती है। यह क्लाइंट प्रोग्राम में TCP सॉकेट बनाकर किया जाता है। जब क्लाइंट अपना TCP सॉकेट बनाता है, तो वह सर्वर में स्वागत सॉकेट का पता निर्दिष्ट करता है, अर्थात, सर्वर होस्ट का IP पता और सॉकेट का पोर्ट नंबर। अपना सॉकेट बनाने के बाद, क्लाइंट तीन-तरफ़ा हैंडशेक आरंभ करता है और सर्वर के साथ TCP कनेक्शन स्थापित करता है। तीन-तरफ़ा हैंडशेक, जो सर्वर के भीतर होता है

ट्रांसपोर्ट लेयर, क्लाइंट और सर्वर प्रोग्रामों के लिए पूरी तरह से अदृश्य है।

तीन-तरफ़ा हैंडशेक के दौरान, क्लाइंट प्रक्रिया सर्वर के स्वागत द्वार पर दस्तक देती है

प्रक्रिया। जब सर्वर दस्तक सुनता है, तो यह एक नया दरवाज़ा बनाता है - अधिक सटीक रूप से, एक नया सॉकेट जो उस विशेष क्लाइंट को समर्पित होता है। नीचे दिए गए हमारे उदाहरण में, स्वागत द्वार एक TCP सॉकेट है

ऑब्जेक्ट जिसे हम serverSocket **कहते हैं** ; क्लाइंट को समर्पित नव निर्मित सॉकेट जो सर्वर सॉकेट बनाता है।

कनेक्शन को connectionSocket **कहा जाता है** । जो छात्र पहली बार टीसीपी सॉकेट का सामना कर रहे हैं, वे कभी-कभी स्वागत सॉकेट (जो सर्वर के साथ संचार करने के इच्छुक सभी क्लाइंट के लिए संपर्क का प्रारंभिक बिंदु है) और प्रत्येक नए बनाए गए सर्वर-साइड कनेक्शन सॉकेट को भ्रमित करते हैं जो बाद में प्रत्येक क्लाइंट के साथ संचार करने के लिए बनाया जाता है।

अनुप्रयोग के परिप्रेक्ष्य से, क्लाइंट का सॉकेट और सर्वर का कनेक्शन सॉकेट सीधे जुड़े हुए हैं

पाइप द्वारा जुड़ा हुआ है। जैसा कि चित्र 2.28 में दिखाया गया है, क्लाइंट प्रक्रिया अपने सॉकेट में मनमाने ढंग से बाइट्स भेज सकती है, और TCP गारंटी देता है कि सर्वर प्रक्रिया (कनेक्शन सॉकेट के माध्यम से) भेजे गए क्रम में प्रत्येक बाइट प्राप्त करेगी। इस प्रकार TCP क्लाइंट और सर्वर प्रक्रियाओं के बीच एक विश्वसनीय सेवा प्रदान करता है।

इसके अलावा, जिस तरह लोग एक ही दरवाजे से अंदर और बाहर आ-जा सकते हैं, क्लाइंट प्रक्रिया न केवल बाइट्स भेजती है

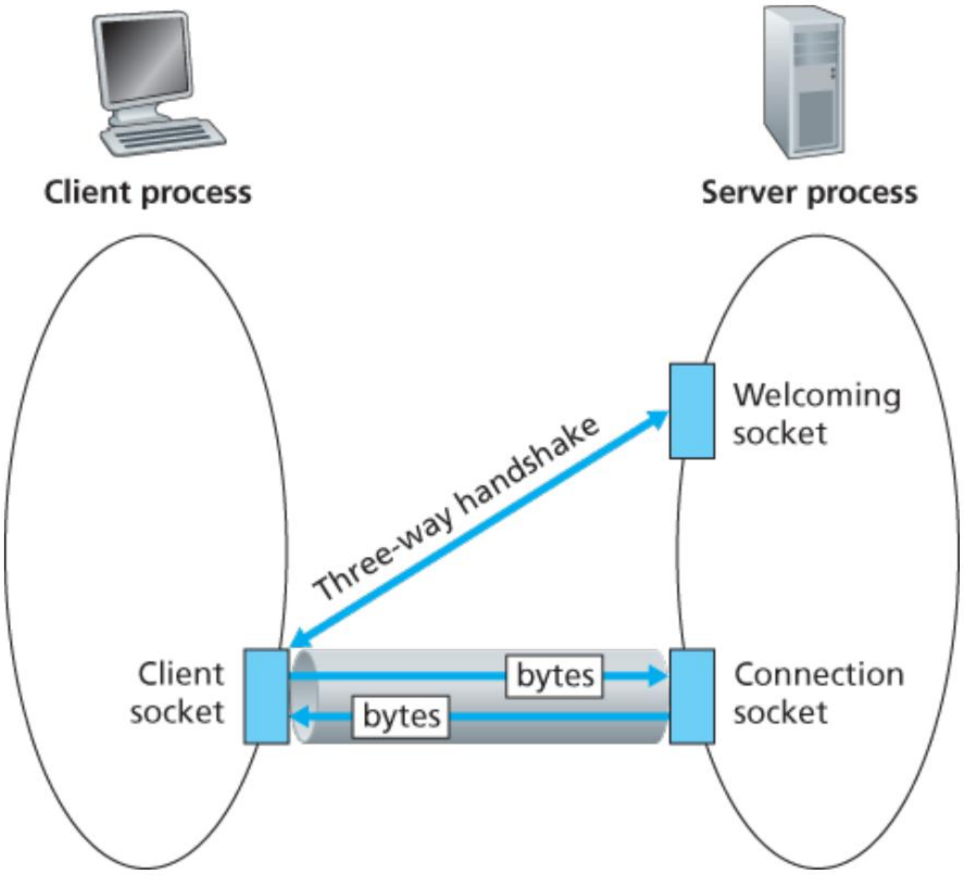
न केवल अपने सॉकेट से बाइट्स प्राप्त करता है, बल्कि सर्वर प्रक्रिया भी अपने सॉकेट से बाइट्स प्राप्त करती है।

इसके कनेक्शन सॉकेट में बाइट्स भी भेजता है।

हम TCP के साथ सॉकेट प्रोग्रामिंग को प्रदर्शित करने के लिए उसी सरल क्लाइंट-सर्वर अनुप्रयोग का उपयोग करते हैं: क्लाइंट सर्वर को डेटा की एक पंक्ति भेजता है, सर्वर उस पंक्ति को कैपिटल में लिखकर उसे वापस क्लाइंट को भेज देता है।

चित्र 2.29 क्लाइंट और सर्वर की मुख्य सॉकेट-संबंधी गतिविधि पर प्रकाश डालता है जो संचार करते हैं

टीसीपी परिवहन सेवा.



चित्र 2.28 TCPServer प्रक्रिया में दो सॉकेट हैं

TCPClient.py

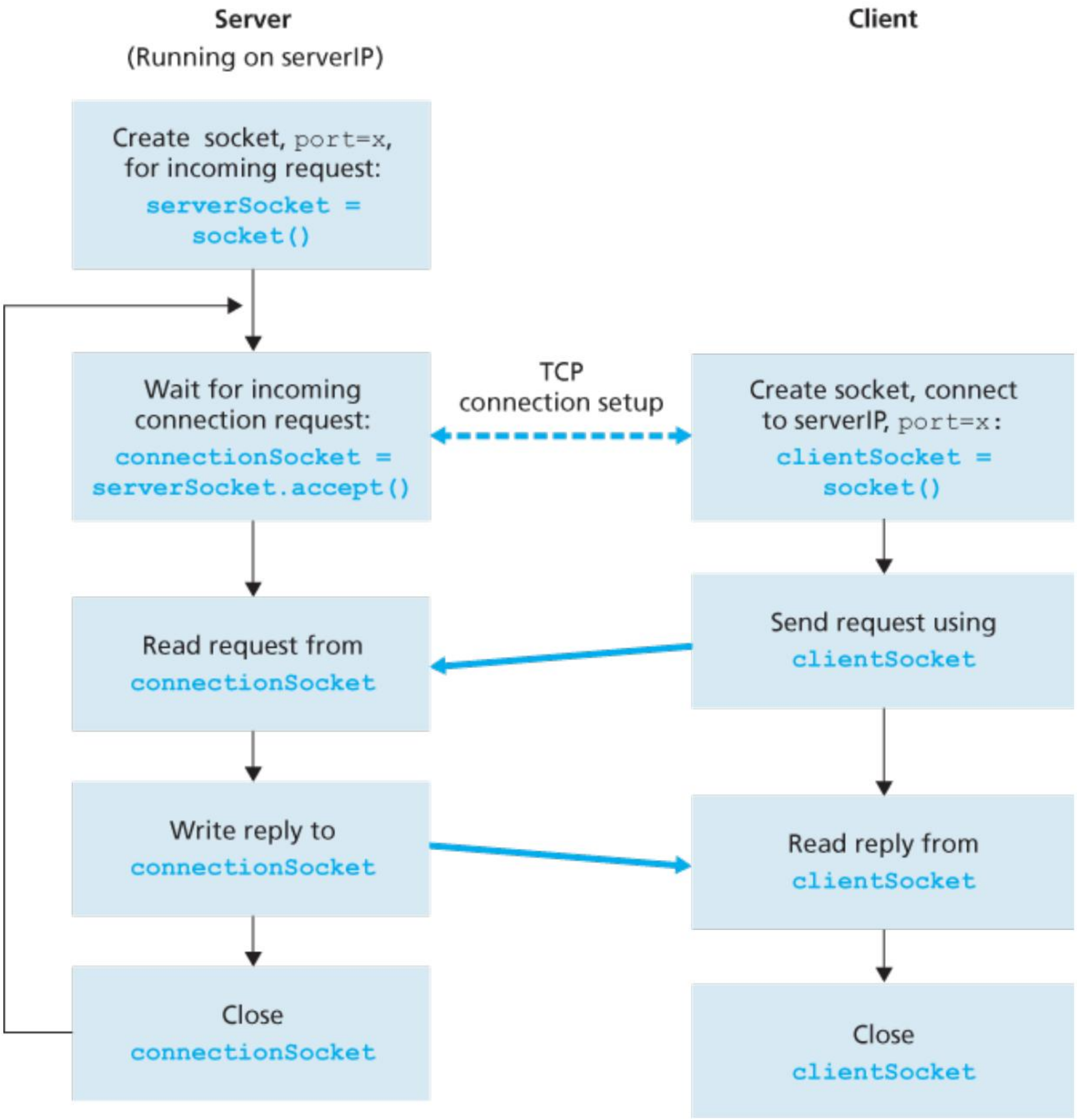
एप्लिकेशन के क्लाइंट साइड का कोड यहां दिया गया है:

```
सॉकेट से आयात करें *
सर्वरनाम = 'सर्वरनाम'
सर्वरपोर्ट = 12000
क्लाइंट सॉकेट = सॉकेट (AF_INET, SOCK_STREAM)
clientSocket.connect((सर्वरनाम, सर्वरपोर्ट))
वाक्य = raw_input('इनपुट लोअरकेस वाक्य:')
clientSocket.send(वाक्य.encode())
संशोधित वाक्य = clientSocket.recv(1024)
प्रिंट('सर्वर से: ', संशोधित वाक्य.डिकोड())
क्लाइंट सॉकेट.क्लोज()
```

आइए अब कोड में विभिन्न पंक्तियों पर नज़र डालें जो UDP कार्यान्वयन से काफी भिन्न हैं। ऐसी पहली पंक्ति क्लाइंट सॉकेट का निर्माण है।

```
क्लाइंट सॉकेट = सॉकेट (AF_INET, SOCK_STREAM)
```

यह लाइन क्लाइंट सॉकेट बनाती है, जिसे clientSocket **कहा जाता है** । पहला पैरामीटर फिर से संकेत देता है कि अंतर्निहित नेटवर्क IPv4 का उपयोग कर रहा है। दूसरा पैरामीटर



चित्र 2.29 TCP का उपयोग करने वाला क्लाइंट-सर्वर अनुप्रयोग

यह दर्शाता है कि सॉकेट SOCK_STREAM **प्रकार का है ,** जिसका अर्थ है कि यह एक TCP सॉकेट है (UDP सॉकेट के बजाय)। ध्यान दें कि जब हम इसे बनाते हैं तो हम क्लाइंट सॉकेट का पोर्ट नंबर निर्दिष्ट नहीं कर रहे हैं; इसके बजाय हम ऑपरेटिंग सिस्टम को यह हमारे लिए करने दे रहे हैं। अब कोड की अगली पंक्ति इससे बहुत अलग है

हमने UDPClient में क्या देखा:

```
clientSocket.connect((सर्वरनाम, सर्वरपोर्ट))
```

स्मरण रखें कि इससे पहले कि क्लाइंट TCP सॉकेट का उपयोग करके सर्वर को डेटा भेज सके (या इसके विपरीत), एक TCP
सबसे पहले क्लाइंट और सर्वर के बीच कनेक्शन स्थापित होना चाहिए। उपरोक्त लाइन TCP आरंभ करती है

क्लाइंट और सर्वर के बीच कनेक्शन। कनेक्ट() विधि का पैरामीटर कनेक्शन के सर्वर साइड का पता है। कोड की इस लाइन के निष्पादित होने के बाद, तीन-तरफ़ा हैंडशेक किया जाता है और क्लाइंट और सर्वर के बीच एक TCP कनेक्शन स्थापित किया जाता है।

```
वाक्य = raw_input('इनपुट लोअरकेस वाक्य:')
```

UDPClient की तरह, ऊपर दिया गया वाक्य उपयोगकर्ता से एक वाक्य प्राप्त करता है। स्ट्रिंग वाक्य तब तक अक्षर इकट्ठा करता रहता है जब तक उपयोगकर्ता कैरिज रिटर्न टाइप करके लाइन समाप्त नहीं कर देता। कोड की अगली पंक्ति भी UDPClient से बहुत अलग है:

```
clientSocket.send(वाक्य.encode())
```

उपरोक्त पंक्ति क्लाइंट के सॉकेट के माध्यम से वाक्य को TCP कनेक्शन में भेजती है। ध्यान दें कि

प्रोग्राम स्पष्ट रूप से पैकेट नहीं बनाता है और गंतव्य पते को पैकेट से नहीं जोड़ता है, जैसा कि UDP सॉकेट के मामले में था। इसके बजाय क्लाइंट प्रोग्राम केवल स्ट्रिंग वाक्य में बाइट्स को TCP कनेक्शन में छोड़ देता है। क्लाइंट तब सर्वर से बाइट्स प्राप्त करने की प्रतीक्षा करता है।

```
संशोधित वाक्य = clientSocket.recv(2048)
```

जब अक्षर सर्वर से आते हैं, तो उन्हें संशोधित वाक्य स्ट्रिंग में रखा जाता है ।

संशोधित वाक्य में अक्षर तब तक जमा होते रहते हैं जब तक कि लाइन कैरिज रिटर्न कैरेक्टर के साथ समाप्त नहीं हो जाती। कैपिटलाइज़्ड वाक्य को प्रिंट करने के बाद, हम क्लाइंट के सॉकेट को बंद कर देते हैं:

```
क्लाइंट सॉकेट.क्लोज()
```

यह अंतिम पंक्ति सॉकेट को बंद कर देती है और इस प्रकार, क्लाइंट और सर्वर के बीच TCP कनेक्शन को बंद कर देती है।

यह क्लाइंट में TCP को सर्वर में TCP को एक TCP संदेश भेजने का कारण बनता है (अनुभाग 3.5 देखें)।

TCPServer.py

अब आइये सर्वर प्रोग्राम पर नजर डालें।

```
सॉकेट से आयात करें *
सर्वरपोर्ट = 12000
सर्वर सॉकेट = सॉकेट (AF_INET, SOCK_STREAM)
सर्वर सॉकेट.बाइंड(('', सर्वर पोर्ट))
सर्वर सॉकेट.listen(1)
प्रिंट ('सर्वर प्राप्त करने के लिए तैयार है')
जबकि सत्य:
    कनेक्शन सॉकेट, addr = सर्वर सॉकेट.स्वीकार करें()
    वाक्य = कनेक्शन सॉकेट.recv(1024).decode()
    capitalizedSentence = वाक्य.upper()
    कनेक्शन सॉकेट.send(capitalizedSentence.encode())
    कनेक्शन सॉकेट.बंद करें()
```

आइए अब उन लाइनों पर नज़र डालें जो UDPServer और TCPClient से काफ़ी अलग हैं। TCPClient, सर्वर एक TCP सॉकेट बनाता है:

```
सर्वरसॉकेट=सॉकेट(AF_INET, SOCK_STREAM)
```

UDPServer के समान, हम सर्वर पोर्ट नंबर, serverPort को जोड़ते हैं , इस सॉकेट के साथ:

```
सर्वर सॉकेट.बाइंड(('', सर्वर पोर्ट))
```

लेकिन TCP के साथ, serverSocket हमारा स्वागत करने वाला सॉकेट होगा। इस स्वागत द्वार को स्थापित करने के बाद, हम किसी ग्राहक के दरवाजे पर दस्तक देने की प्रतीक्षा करेंगे और सुनेंगे:

```
सर्वर सॉकेट.listen(1)
```

इस लाइन में सर्वर क्लाइंट से TCP कनेक्शन अनुरोधों को सुनता है। पैरामीटर कतारबद्ध कनेक्शनों की अधिकतम संख्या (कम से कम 1) निर्दिष्ट करता है।

```
कनेक्शन सॉकेट, addr = सर्वर सॉकेट.स्वीकार करें()
```

जब कोई क्लाइंट इस दरवाजे पर दस्तक देता है, तो प्रोग्राम serverSocket के लिए Accept() विधि को लागू करता है, जो सर्वर में एक नया सॉकेट बनाता है, जिसे कनेक्शन सॉकेट कहा जाता है , इस विशेष ग्राहक को समर्पित. इसके बाद क्लाइंट और सर्वर हैंडशेकिंग पूरी करते हैं, जिससे क्लाइंट और सर्वर के बीच एक TCP कनेक्शन बनता है। clientSocket और सर्वर का connectionSocket । TCP कनेक्शन स्थापित होने के साथ, क्लाइंट और सर्वर अब कनेक्शन पर एक दूसरे को बाइट्स भेज सकते हैं। TCP के साथ, क्लाइंट और सर्वर से भेजे गए सभी बाइट्स एक तरफ से आने वाले सामान के न केवल दूसरे तरफ पहुंचने की गारंटी है, बल्कि यह भी गारंटी है कि वे क्रम में पहुंचेंगे।

```
कनेक्शन सॉकेट.बंद करें()
```

इस प्रोग्राम में, संशोधित वाक्य को क्लाइंट को भेजने के बाद, हम कनेक्शन सॉकेट को बंद कर देते हैं। लेकिन चूंकि सर्वर सॉकेट खुला रहता है, इसलिए अब कोई अन्य क्लाइंट दरवाजा खटखटा सकता है और सर्वर को संशोधित करने के लिए कोई वाक्य भेज सकता है।

यह TCP में सॉकेट प्रोग्रामिंग पर हमारी चर्चा पूरी करता है। आपको दो अलग-अलग होस्ट में दो प्रोग्राम चलाने के लिए प्रोत्साहित किया जाता है, और थोड़े अलग लक्ष्य प्राप्त करने के लिए उन्हें संशोधित भी करना चाहिए। आपको UDP प्रोग्राम जोड़ी की तुलना TCP प्रोग्राम जोड़ी से करनी चाहिए और देखना चाहिए कि वे कैसे भिन्न हैं। आपको यह भी करना चाहिए

अध्याय 2, 4, और 9 के अंत में वर्णित कई सॉकेट प्रोग्रामिंग असाइनमेंट। अंत में, हम आशा करते हैं कि किसी दिन, इन और अधिक उन्नत सॉकेट कार्यक्रमों में महारत हासिल करने के बाद, आप अपना खुद का लोकप्रिय नेटवर्क एप्लिकेशन लिखेंगे, बहुत अमीर और प्रसिद्ध हो जाएंगे, और इस पाठ्यपुस्तक के लेखकों को याद रखेंगे!

2.8 सारांश

इस अध्याय में हमने नेटवर्क अनुप्रयोगों के संकल्पनात्मक और कार्यान्वयन पहलुओं का अध्ययन किया है। हमने कई इंटरनेट अनुप्रयोगों द्वारा अपनाए गए सर्वव्यापी क्लाइंट-सर्वर आर्किटेक्चर के बारे में सीखा है और HTTP, SMTP, POP3 और DNS प्रोटोकॉल में इसके उपयोग को देखा है। हमने इन महत्वपूर्ण एप्लिकेशन-स्तरीय प्रोटोकॉल और उनके संबंधित संबद्ध अनुप्रयोगों (वेब, फ़ाइल ट्रांसफ़र, ई-मेल और DNS) का कुछ विस्तार से अध्ययन किया है। हमने P2P आर्किटेक्चर के बारे में सीखा है और यह कई अनुप्रयोगों में कैसे उपयोग किया जाता है।

हमने स्ट्रीमिंग वीडियो के बारे में भी सीखा तथा यह भी सीखा कि आधुनिक वीडियो वितरण प्रणालियां CDN का किस प्रकार लाभ उठाती हैं। हमने जांच की है कि नेटवर्क एप्लिकेशन बनाने के लिए सॉकेट API का उपयोग कैसे किया जा सकता है। हमने कनेक्शन-ओरिएंटेड (TCP) और कनेक्शनलेस (UDP) एंड-टू-एंड ट्रांसपोर्ट सेवाओं के लिए सॉकेट के उपयोग के बारे में जाना है। लेयर्ड नेटवर्क आर्किटेक्चर की हमारी यात्रा का पहला चरण अब पूरा हो गया है!

इस पुस्तक के आरंभ में, अनुभाग 1.1 में, हमने प्रोटोकॉल की एक अस्पष्ट, सरल परिभाषा दी है: "दो या अधिक संचार करने वाली संस्थाओं के बीच आदान-प्रदान किए जाने वाले संदेशों का प्रारूप और क्रम, साथ ही संदेश या अन्य घटना के प्रेषण और/या प्राप्ति पर की जाने वाली कार्रवाइयां।"

इस अध्याय की सामग्री, और विशेष रूप से HTTP, SMTP, POP3, और DNS प्रोटोकॉल के हमारे विस्तृत अध्ययन ने अब इस परिभाषा में काफी सामग्री जोड़ दी है। प्रोटोकॉल नेटवर्किंग में एक महत्वपूर्ण अवधारणा है; एप्लिकेशन प्रोटोकॉल के हमारे अध्ययन ने अब हमें प्रोटोकॉल के बारे में अधिक सहज ज्ञान विकसित करने का अवसर दिया है।

अनुभाग 2.1 में , हमने उन सेवा मॉडलों का वर्णन किया है जो TCP और UDP उन अनुप्रयोगों को प्रदान करते हैं जो उन्हें लागू करते हैं। हमने इन सेवा मॉडलों पर और भी करीब से नज़र डाली जब हमने सरल अनुप्रयोग विकसित किए

अनुभाग 2.7 में TCP और UDP पर चर्चा की गई है । हालाँकि, हमने इस बारे में बहुत कम कहा है कि TCP और UDP ये सेवा मॉडल कैसे प्रदान करते हैं। उदाहरण के लिए, हम जानते हैं कि TCP एक विश्वसनीय डेटा सेवा प्रदान करता है, लेकिन हमने नहीं किया है

यह कहा तो नहीं गया, लेकिन यह ऐसा कैसे करता है। अगले अध्याय में हम न केवल परिवहन प्रोटोकॉल के बारे में बल्कि इसके कैसे और क्यों पर भी ध्यान से नज़र डालेंगे ।

इंटरनेट अनुप्रयोग संरचना और अनुप्रयोग-स्तरीय प्रोटोकॉल के बारे में ज्ञान से लैस होकर, अब हम

अध्याय 3 में प्रोटोकॉल स्टैक में आगे बढ़ने और ट्रांसपोर्ट लेयर की जांच करने के लिए तैयार रहें ।

गृहकार्य की समस्याएं और प्रश्न

अध्याय 2 समीक्षा प्रश्न

अनुभाग 2.1

R1. पाँच गैर-स्वामित्व वाले इंटरनेट अनुप्रयोगों और उनके अनुप्रयोग-स्तर प्रोटोकॉल की सूची बनाएँ
उपयोग।

R2. नेटवर्क आर्किटेक्चर और एप्लिकेशन आर्किटेक्चर के बीच क्या अंतर है?

R3. प्रक्रियाओं की एक जोड़ी के बीच संचार सत्र के लिए, कौन सी प्रक्रिया क्लाइंट है और कौन सी सर्वर है?

R4. एक P2P फ़ाइल-शेयरिंग एप्लिकेशन के लिए, क्या आप इस कथन से सहमत हैं, "संचार सत्र के क्लाइंट और सर्वर पक्षों की कोई धारणा नहीं है"? क्यों या क्यों नहीं?

R5. एक होस्ट पर चल रही प्रक्रिया द्वारा दूसरे होस्ट पर चल रही प्रक्रिया की पहचान करने के लिए कौन सी जानकारी का उपयोग किया जाता है?

R6. मान लीजिए कि आप किसी दूरस्थ क्लाइंट से सर्वर तक यथाशीघ्र लेनदेन करना चाहते हैं।
क्या आप UDP या TCP का उपयोग करेंगे? क्यों?

R7. चित्र 2.4 **का संदर्भ लें ,** जिसमें डेटा हानि **, हम देखते हैं कि** चित्र 2.4 में सूचीबद्ध किसी भी अनुप्रयोग को इसकी आवश्यकता नहीं है
और समय दोनों नहीं हैं। क्या आप ऐसे अनुप्रयोग की कल्पना कर सकते हैं जिसमें डेटा हानि की आवश्यकता न हो और जो अत्यधिक समय-संवेदनशील भी हो?

R8. ट्रांसपोर्ट प्रोटोकॉल द्वारा प्रदान की जा सकने वाली सेवाओं की चार व्यापक श्रेणियों की सूची बनाएँ। प्रत्येक सेवा श्रेणी के लिए, इंगित करें कि क्या UDP या TCP (या दोनों) ऐसी सेवा प्रदान करते हैं।

R9. याद रखें कि एन्क्रिप्शन सहित प्रक्रिया-दर-प्रक्रिया सुरक्षा सेवाएँ प्रदान करने के लिए TCP को SSL के साथ बढ़ाया जा सकता है। क्या SSL ट्रांसपोर्ट लेयर या एप्लिकेशन लेयर पर काम करता है? यदि एप्लिकेशन डेवलपर चाहता है कि TCP को SSL के साथ बढ़ाया जाए, तो डेवलपर को क्या करना होगा

करना?

अनुभाग 2.2–2.5

R10. हैंडशेकिंग प्रोटोकॉल से क्या अभिप्राय है?

R11. HTTP, SMTP और POP3 UDP के बजाय TCP के ऊपर क्यों चलते हैं?

R12. एक ई-कॉमर्स साइट पर विचार करें जो अपने प्रत्येक ग्राहक के लिए खरीदारी का रिकॉर्ड रखना चाहती है। बताएं कि कुकीज़ के साथ ऐसा कैसे किया जा सकता है।

R13. बताएं कि वेब कैशिंग अनुरोधित ऑब्जेक्ट प्राप्त करने में होने वाली देरी को कैसे कम कर सकता है। क्या वेब कैशिंग उपयोगकर्ता द्वारा अनुरोधित सभी ऑब्जेक्ट के लिए देरी को कम करेगा या केवल कुछ ऑब्जेक्ट के लिए?

क्यों?

R14. वेब सर्वर में टेलनेट करें और एक मल्टीलाइन अनुरोध संदेश भेजें। अनुरोध संदेश में If-modified-since: हेडर लाइन शामिल करें ताकि 304 के साथ प्रतिक्रिया संदेश को बाध्य किया जा सके

संशोधित स्थिति कोड नहीं.

R15. कई लोकप्रिय मैसेजिंग ऐप्स की सूची बनाएं। क्या वे एसएमएस के समान प्रोटोकॉल का उपयोग करते हैं?

R16. मान लीजिए कि एलिस, वेब-आधारित ई-मेल खाते (जैसे हॉटमेल या जीमेल) के साथ, बॉब को एक संदेश भेजती है, जो POP3 का उपयोग करके अपने मेल सर्वर से अपने मेल तक पहुँचता है। चर्चा करें कि एलिस के होस्ट से बॉब के होस्ट तक संदेश कैसे पहुँचता है। एप्लिकेशन-लेयर की श्रृंखला को सूचीबद्ध करना सुनिश्चित करें

प्रोटोकॉल जो दो होस्टों के बीच संदेश को स्थानांतरित करने के लिए उपयोग किए जाते हैं।

R17. हाल ही में आपको प्राप्त हुए ई-मेल संदेश का हेडर प्रिंट करें। इसमें कितनी Received: हेडर लाइनें हैं? संदेश में प्रत्येक हेडर लाइन का विश्लेषण करें।

R18. उपयोगकर्ता के दृष्टिकोण से, POP3 में डाउनलोड-एवं-डिलीट मोड और डाउनलोड-एवं-कीप मोड के बीच क्या अंतर है?

R19. क्या किसी संगठन के वेब सर्वर और मेल सर्वर के लिए होस्टनाम के लिए बिल्कुल समान उपनाम रखना संभव है (उदाहरण के लिए, foo.com)? RR का प्रकार क्या होगा जिसमें शामिल है

मेल सर्वर का होस्टनाम?

R20. अपने प्राप्त ईमेल देखें, और .edu ईमेल पते वाले उपयोगकर्ता से भेजे गए संदेश के हेडर की जांच करें। क्या हेडर से होस्ट का आईपी पता निर्धारित करना संभव है

संदेश किससे भेजा गया था? जीमेल खाते से भेजे गए संदेश के लिए भी ऐसा ही करें।

अनुभाग 2.5

R21. बिटटोरेंट में, मान लीजिए कि ऐलिस 30 सेकंड के अंतराल में बॉब को चंक प्रदान करती है। क्या बॉब भी उसी अंतराल में ऐलिस को चंक प्रदान करेगा? क्यों या क्यों

नहीं?

R22. एक नए साथी एलिस पर विचार करें जो बिना किसी चंक के बिटटोरेंट में शामिल होता है। बिना किसी चंक के, वह किसी भी अन्य साथी के लिए शीर्ष-चार अपलोडर नहीं बन सकता, क्योंकि उसके पास अपलोड करने के लिए कुछ भी नहीं है। फिर एलिस को अपना पहला चंक कैसे मिलेगा?

R23. ओवरले नेटवर्क क्या है? क्या इसमें राउटर शामिल हैं? ओवरले नेटवर्क में किनारे क्या हैं?

अनुभाग 2.6

R24. CDN आमतौर पर दो अलग-अलग सर्वर प्लेसमेंट दर्शन में से एक को अपनाते हैं। उनका नाम बताएँ और उनका संक्षिप्त वर्णन करें।

R25. नेटवर्क से संबंधित विचारों जैसे कि देरी, हानि और बैंडविड्थ प्रदर्शन के अलावा, अन्य महत्वपूर्ण कारक भी हैं जो CDN सर्वर चयन रणनीति को डिजाइन करने में शामिल होते हैं। वे क्या हैं?

अनुभाग 2.7

R26. अनुभाग 2.7 में वर्णित UDP सर्वर को केवल एक सॉकेट की आवश्यकता थी, जबकि TCP सर्वर को दो सॉकेट की आवश्यकता थी। क्यों? यदि TCP सर्वर को n समकालिक कनेक्शनों का समर्थन करना था, जिनमें से प्रत्येक एक अलग क्लाइंट होस्ट से था, तो TCP सर्वर को कितने सॉकेट की आवश्यकता होगी?

R27. धारा 2.7 में वर्णित TCP पर क्लाइंट-सर्वर अनुप्रयोग के लिए , सर्वर प्रोग्राम को क्लाइंट प्रोग्राम से पहले क्यों निष्पादित किया जाना चाहिए? UDP पर क्लाइंट-सर्वर अनुप्रयोग के लिए, क्लाइंट प्रोग्राम को सर्वर प्रोग्राम से पहले क्यों निष्पादित किया जा सकता है?

समस्याएं

P1. सत्य या असत्य?

a. कोई उपयोगकर्ता एक वेब पेज का अनुरोध करता है जिसमें कुछ पाठ और तीन छवियाँ होती हैं। इस पेज के लिए, क्लाइंट एक अनुरोध संदेश भेजेगा और चार प्रतिक्रिया संदेश प्राप्त करेगा।

b. दो अलग-अलग वेब पेज (उदाहरण के लिए, www.mit.edu/research.html और www.mit.edu/students.html) को उसी स्थायी कनेक्शन पर भेजा जा सकता है।

सी. ब्राउज़र और मूल सर्वर के बीच गैर-स्थायी कनेक्शन के साथ, यह संभव है दो अलग-अलग HTTP अनुरोध संदेशों को ले जाने के लिए एकल TCP खंड का उपयोग किया जाता है।

d. HTTP **प्रतिक्रिया संदेश में** दिनांक : हेडर यह इंगित करता है कि ऑब्जेक्ट कब प्रतिक्रिया अंतिम बार संशोधित की गई थी।

e. HTTP प्रतिक्रिया संदेशों में कभी भी खाली संदेश निकाय नहीं होता है।

P2. SMS, iMessage और WhatsApp सभी स्मार्टफोन रियल-टाइम मैसेजिंग सिस्टम हैं। इंटरनेट पर कुछ शोध करने के बाद, इनमें से प्रत्येक सिस्टम के लिए उनके द्वारा उपयोग किए जाने वाले प्रोटोकॉल के बारे में एक पैराग्राफ लिखें। फिर एक पैराग्राफ लिखें जिसमें बताया जाए कि वे कैसे भिन्न हैं।

P3. एक HTTP क्लाइंट पर विचार करें जो किसी दिए गए URL पर एक वेब दस्तावेज़ प्राप्त करना चाहता है। HTTP सर्वर का IP पता शुरू में अज्ञात है। कौन से ट्रांसपोर्ट और एप्लिकेशन-लेयर प्रोटोकॉल इस परिदृश्य में HTTP के अलावा और क्या आवश्यक है?

P4. ASCII वर्णों की निम्नलिखित स्ट्रिंग पर विचार करें, जिसे Wireshark द्वारा तब कैप्चर किया गया था जब ब्राउज़र ने HTTP GET संदेश भेजा था (अर्थात, यह HTTP GET संदेश की वास्तविक सामग्री है)।

<cr><lf> अक्षर कैरिज रिटर्न और लाइन-फीड अक्षर हैं (अर्थात, नीचे दिए गए पाठ में इटैलाइज्ड कैरेक्टर स्ट्रिंग <cr> एकल कैरिज-रिटर्न कैरेक्टर को दर्शाता है जो HTTP हेडर में उस बिंदु पर मौजूद था)। नीचे दिए गए HTTP GET संदेश में आपको उत्तर कहां मिलेगा, यह इंगित करते हुए निम्नलिखित प्रश्नों के उत्तर दें।

```
GET /cs453/index.html HTTP/1.1<cr><lf>होस्ट: gai
a.cs.umass.edu<cr><lf>उपयोगकर्ता-एजेंट: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 नेटस्केप/7.2 (ax) <cr><lf>स्वीकार करें:ex
```

```
t/xml, एप्लीकेशन/xml, एप्लीकेशन/xhtml+xml, पाठ
/html;q=0.9, टेक्स्ट/प्लेन;q=0.8, इमेज/png,*/*;q=0.5
<cr><lf>स्वीकार-भाषा: en-us, en;q=0.5<cr><lf>स्वीकार-
एनकोडिंग: ज़िप, डिफ्लेट<cr><lf>स्वीकार-चरित्र: ISO
-8859-1, utf-8;q=0.7,*;q=0.7<cr><lf>कीप-अलाइव: 300<cr>
<lf>कनेक्शन:keep-alive<cr><lf><cr><lf>
```

a. ब्राउज़र द्वारा अनुरोधित दस्तावेज़ का URL क्या है? b. ब्राउज़र HTTP का कौन सा संस्करण चला रहा है? c. क्या ब्राउज़र एक गैर-स्थायी या एक स्थायी कनेक्शन का अनुरोध करता है? d. उस होस्ट का IP पता क्या है जिस पर ब्राउज़र चल रहा है? e. किस प्रकार का ब्राउज़र इस संदेश को आरंभ करता है? HTTP अनुरोध संदेश में ब्राउज़र प्रकार की आवश्यकता क्यों है?

P5. नीचे दिया गया पाठ ऊपर दिए गए प्रश्न में HTTP GET संदेश के जवाब में सर्वर से भेजा गया उत्तर दिखाता है। नीचे दिए गए प्रश्नों के उत्तर दें, यह इंगित करते हुए कि नीचे दिए गए संदेश में आपको उत्तर कहाँ मिला।

```
HTTP/1.1 200 OK<cr><lf>दिनांक: मंगलवार, 07 मार्च 2008
12:39:45GMT<cr><lf>सर्वर: Apache/2.0.52 (Fedora)
<cr><lf>अंतिम-संशोधन: शनि, 10 दिसम्बर 2005 18:27:46
GMT<cr><lf>ETag: ”526c3-f22-a88a4c80”<cr><lf>स्वीकार करें-
श्रेणियाँ: बाइट्स<cr><lf>सामग्री-लंबाई: 3874<cr><lf>
कीप-अलाइव: टाइमआउट=अधिकतम=100<cr><lf>कनेक्शन:
Keep-Alive<cr><lf>सामग्री-प्रकार: text/html; charset=
ISO-8859-1<cr><lf><cr><lf><!doctype html public ”-
//w3c//dtd html 4.0 संक्रमणकालीन//en”><lf><html><lf>
<head><lf> <meta http-equiv="Content-Type"
सामग्री = "टेक्स्ट / html; वर्णसेट = iso-8859-1"> <lf> <मेटा
नाम = "जेनरेटर" सामग्री = "मोज़िला / 4.79 [en] (विंडोज एनटी
5.0; यू) नेटस्केप]”><lf> <title>सीएमपीएससीआई 453 / 591 /
NTU-ST550Aस्प्रिंग 2005 होमपेज</title><lf></head><lf>
<अधिक दस्तावेज़ पाठ यहाँ दिया गया है (नहीं दिखाया गया है)>
```

a. क्या सर्वर दस्तावेज़ को सफलतापूर्वक ढूँढ़ पाया या नहीं?
दस्तावेज़ उत्तर प्रदान किया गया?

ख. दस्तावेज़ को अंतिम बार कब संशोधित किया गया था?

c. लौटाए जा रहे दस्तावेज़ में कितने बाइट्स हैं? d. लौटाए जा रहे दस्तावेज़ के पहले 5 बाइट्स क्या हैं? क्या सर्वर किसी भी तरह से सहमत था?

निरंतर कनेक्शन?

P6. HTTP/1.1 विनिर्देश (RFC 2616) प्राप्त करें। निम्नलिखित प्रश्नों के उत्तर दें:

a. क्लाइंट और सर्वर के बीच सिग्नलिंग के लिए इस्तेमाल की जाने वाली प्रणाली को समझाइए, जिससे यह संकेत मिलता है कि कोई स्थायी कनेक्शन बंद किया जा रहा है। क्या क्लाइंट, सर्वर या दोनों ही बंद होने का संकेत दे सकते हैं एक कनेक्शन का?

बी. HTTP द्वारा कौन सी एन्क्रिप्शन सेवाएँ प्रदान की जाती हैं? सी. क्या कोई क्लाइंट किसी दिए गए सर्वर के साथ एक साथ तीन या उससे ज़्यादा कनेक्शन खोल सकता है? डी. अगर कोई सर्वर या क्लाइंट यह पता लगाता है कि कनेक्शन कुछ समय से निष्क्रिय है, तो वह अपने बीच ट्रांसपोर्ट कनेक्शन बंद कर सकता है। क्या यह संभव है कि एक पक्ष कनेक्शन बंद करना शुरू कर दे, जबकि दूसरा पक्ष इस कनेक्शन के ज़रिए डेटा ट्रांसमिट कर रहा हो?

व्याख्या करना।

P7. मान लीजिए कि आप अपने वेब ब्राउज़र में किसी वेब पेज को प्राप्त करने के लिए किसी लिंक पर क्लिक करते हैं। संबंधित URL का IP पता आपके स्थानीय होस्ट में कैश नहीं है, इसलिए इसे प्राप्त करने के लिए DNS लुकअप आवश्यक है।

आईपी पता। मान लीजिए कि आपके होस्ट को DNS से आईपी पता प्राप्त होने से पहले n **DNS सर्वर का दौरा किया जाता है; लगातार विज़िट में** RTT1,. . .,RTTn **का** RTT **होता है।** इसके अलावा मान लीजिए कि लिंक से जुड़े वेब पेज में केवल एक ऑब्जेक्ट है, जिसमें थोड़ी मात्रा में HTML शामिल है

टेक्स्ट: RTT को स्थानीय0होस्ट और ऑब्जेक्ट वाले सर्वर के बीच RTT को दर्शाएँ।

यह मानते हुए कि ऑब्जेक्ट का ट्रांसमिशन समय शून्य है, क्लाइंट द्वारा लिंक पर क्लिक करने से लेकर क्लाइंट को ऑब्जेक्ट प्राप्त होने तक कितना समय बीतता है?

P8. समस्या P7 का संदर्भ देते हुए, मान लीजिए कि HTML फ़ाइल एक ही सर्वर पर आठ बहुत छोटी वस्तुओं का संदर्भ देती है। ट्रांसमिशन समय की उपेक्षा करते हुए, कितना समय बीतता है

a. बिना किसी समानांतर TCP कनेक्शन के गैर-स्थायी HTTP? b. 5 समानांतर कनेक्शनों के लिए कॉन्फ़िगर किए गए ब्राउज़र के साथ गैर-स्थायी HTTP?

सी. स्थायी HTTP?

P9. चित्र 2.12 **पर विचार करें** मान लीजिए , जिसके लिए इंटरनेट से जुड़ा एक संस्थागत नेटवर्क है।

कि औसत ऑब्जेक्ट का आकार 850,000 बिट्स है और संस्थान के ब्राउज़र से मूल सर्वर तक औसत अनुरोध दर 16 अनुरोध प्रति सेकंड है। यह भी मान लीजिए कि

वह समय जो एक्सेस लिंक के इंटरनेट साइड पर राउटर द्वारा एक संदेश को अग्रेषित करने से लेकर लगने वाले समय तक होता है

HTTP अनुरोध को प्रतिक्रिया प्राप्त होने तक औसतन तीन सेकंड का समय लगता है (देखें अनुभाग 2.2.5)।

औसत एक्सेस विलंब (यानी, इंटरनेट राउटर से संस्थान राउटर तक की देरी) और औसत इंटरनेट विलंब के योग के रूप में कुल औसत प्रतिक्रिया समय को मॉडल करें। औसत एक्सेस विलंब के लिए, Δ/(1−Δβ) का उपयोग करें, जहाँ Δ एक्सेस लिंक पर ऑब्जेक्ट भेजने के लिए आवश्यक औसत समय है और b एक्सेस लिंक पर ऑब्जेक्ट के आगमन की दर है।

क. कुल औसत प्रतिक्रिया समय ज्ञात करें। ख. अब मान लीजिए कि संस्थागत LAN में कैश स्थापित है। मान लीजिए कि मिस रेट 0.4 है।

कुल प्रतिक्रिया समय ज्ञात करें.

P10. एक छोटे, 10-मीटर लिंक पर विचार करें, जिस पर एक प्रेषक दोनों दिशाओं में 150 बिट्स/सेकंड की दर से संचार कर सकता है। मान लीजिए कि डेटा युक्त पैकेट 100,000 बिट्स लंबे हैं, और पैकेट

केवल नियंत्रण (जैसे, ACK या हैंडशेकिंग) वाले 200 बिट लंबे होते हैं। मान लें कि N समानांतर कनेक्शन में से प्रत्येक को लिंक बैंडविड्थ का 1/N मिलता है। अब HTTP प्रोटोकॉल पर विचार करें, और मान लें कि प्रत्येक डाउनलोड की गई वस्तु 100 Kbits लंबी है, और प्रारंभिक डाउनलोड की गई वस्तु में एक ही प्रेषक से 10 संदर्भित वस्तुएँ हैं। क्या इस मामले में गैर-स्थायी HTTP के समानांतर उदाहरणों के माध्यम से समानांतर डाउनलोड समझ में आएगा? अब स्थायी HTTP पर विचार करें। क्या आप गैर-स्थायी मामले पर महत्वपूर्ण लाभ की उम्मीद करते हैं? अपने उत्तर का औचित्य सिद्ध करें और उसे स्पष्ट करें।

P11. पिछली समस्या में प्रस्तुत परिदृश्य पर विचार करें। अब मान लीजिए कि लिंक को बॉब द्वारा चार अन्य उपयोगकर्ताओं के साथ साझा किया गया है। बॉब गैर-स्थायी HTTP के समानांतर उदाहरणों का उपयोग करता है, और अन्य चार उपयोगकर्ता समानांतर डाउनलोड के बिना गैर-स्थायी HTTP का उपयोग करते हैं।

क. क्या बॉब के समानांतर कनेक्शन उसे वेब पेजों को अधिक तेज़ी से प्राप्त करने में मदद करते हैं? क्यों या क्यों नहीं? ख. यदि सभी पाँच उपयोगकर्ता गैर-स्थायी HTTP के पाँच समानांतर इंस्टेंस खोलते हैं, तो क्या बॉब के समानांतर कनेक्शन अभी भी फायदेमंद होंगे? क्यों या क्यों नहीं?

P12. किसी सर्वर के लिए एक सरल TCP प्रोग्राम लिखें जो क्लाइंट से इनपुट की लाइनें स्वीकार करता है और सर्वर के मानक आउटपुट पर लाइनों को प्रिंट करता है। (आप पाठ में TCPServer.py प्रोग्राम को संशोधित करके ऐसा कर सकते हैं।) अपने प्रोग्राम को संकलित करें और निष्पादित करें। किसी अन्य मशीन पर जिसमें वेब ब्राउज़र हो, ब्राउज़र में प्रॉक्सी सर्वर को उस होस्ट पर सेट करें जो आपका सर्वर प्रोग्राम चला रहा हो; पोर्ट नंबर को भी उचित रूप से कॉन्फ़िगर करें। आपके ब्राउज़र को अब अपने GET अनुरोध संदेश आपके सर्वर को भेजने चाहिए, और आपके सर्वर को अपने मानक आउटपुट पर संदेश प्रदर्शित करने चाहिए। यह निर्धारित करने के लिए इस प्लेटफ़ॉर्म का उपयोग करें कि आपका ब्राउज़र स्थानीय रूप से कैश की गई वस्तुओं के लिए सशर्त GET संदेश उत्पन्न करता है या नहीं।

P13. SMTP में MAIL FROM : **और** मेल संदेश में From : के बीच क्या अंतर है? स्वयं?

P14. SMTP संदेश के मुख्य भाग के अंत को कैसे चिह्नित करता है? HTTP के बारे में क्या? क्या HTTP संदेश के मुख्य भाग के अंत को चिह्नित करने के लिए SMTP जैसी ही विधि का उपयोग कर सकता है? समझाएँ।

P15. SMTP के लिए RFC 5321 पढ़ें। MTA का क्या मतलब है? प्राप्त स्पैम ई-मेल (वास्तविक स्पैम ई-मेल से संशोधित) पर विचार करें। यह मानते हुए कि इस स्पैम ई-मेल का केवल स्रोत ही दुर्भावनापूर्ण है और अन्य सभी होस्ट ईमानदार हैं, उस दुर्भावनापूर्ण होस्ट की पहचान करें जिसने यह स्पैम ई-मेल बनाया है।

```
से - शुक्र नवंबर 07 13:41:30 2008
वापसी पथ: <tennis5@pp33head.com>
प्राप्त: barmail.cs.umass.edu से (barmail.cs.umass.edu
 [128.119.240.3]) cs.umass.edu (8.13.1/8.12.6) द्वारा
 <hg@cs.umass.edu>; शुक्र, 7 नवंबर 2008 13:27:10 -0500
प्राप्त: asusus-4b96 (localhost [127.0.0.1]) से
barmail.cs.umass.edu (स्पैम फ़ायरवॉल) <hg@cs.umass.edu> के लिए; शुक्र, 7
```

नवंबर 2008 13:27:07 -0500 (ईएसटी)

प्राप्त: asusus-4b96 ([58.88.21.177]) से barmail.cs.umass.edu द्वारा

<hg@cs.umass.edu> के लिए; शुक्र, 7 नवम्बर 2008 13:27:07 -0500 (ईएसटी)

प्राप्त: [58.88.21.177] से inbnd55.exchangeddd.com द्वारा; शनि, 8
नवंबर 2008 01:27:07 +0700

प्रेषक: "जॉनी" <tennis5@pp33head.com>

<hg@cs.umass.edu> को भेजें

विषय: अपनी बचत को कैसे सुरक्षित रखें

P16. POP3 RFC, RFC 1939 पढ़ें। UIDL POP3 कमांड का उद्देश्य क्या है?

P17. अपने ई-मेल तक POP3 के माध्यम से पहुंचने पर विचार करें।

a. मान लीजिए आपने अपने POP मेल क्लाइंट को डाउनलोड-एंड-
डिलीट मोड। निम्नलिखित लेनदेन पूरा करें:

सी: सूची
एस: 1 498
एस: 2 912
एस: ।
सी: रिट्र 1
एस: ब्ला ब्ला...
एस: ....ब्ला
एस: ।
?
?

b. मान लीजिए कि आपने अपने POP मेल क्लाइंट को डाउनलोड-एंड-कीप मोड में काम करने के लिए कॉन्फ़िगर किया है। निम्नलिखित लेनदेन पूरा करें:

सी: सूची
एस: 1 498
एस: 2 912
एस: ।
सी: रिट्र 1
एस: ब्ला ब्ला...
एस: ....ब्ला
एस: ।
?

?

सी. मान लीजिए कि आपने अपने POP मेल क्लाइंट को डाउनलोड-एंड-कीप मोड में काम करने के लिए कॉन्फ़िगर किया है। भाग (बी) में अपनी प्रतिलिपि का उपयोग करते हुए, मान लीजिए कि आप संदेश 1 और 2 प्राप्त करते हैं, POP से बाहर निकलते हैं, और फिर पाँच मिनट बाद आप नया ई-मेल प्राप्त करने के लिए फिर से POP तक पहुँचते हैं। मान लीजिए कि पाँच मिनट के अंतराल में आपको कोई नया संदेश नहीं भेजा गया है। इस दूसरे POP सत्र की प्रतिलिपि प्रदान करें।

पी18.

क. whois **डाटाबेस** क्या है ?

ख. दो DNS सर्वरों के नाम प्राप्त करने के लिए इंटरनेट पर विभिन्न whois डेटाबेस का उपयोग करें। बताएं कि आपने कौन से whois डेटाबेस का उपयोग किया।

c. तीन DNS सर्वरों को DNS क्वेरी भेजने के लिए अपने स्थानीय होस्ट पर nslookup का उपयोग करें: आपका स्थानीय DNS सर्वर और वे दो DNS सर्वर जो आपको भाग (b) में मिले हैं। टाइप A, NS और MX रिपोर्ट के लिए क्वेरी करने का प्रयास करें। अपने निष्कर्षों का सारांश दें।

d. एक वेब सर्वर को खोजने के लिए nslookup का उपयोग करें जिसमें एकाधिक IP पते हों। क्या वेब सर्वर क्या आपके संस्थान (स्कूल या कंपनी) के पास एक से अधिक IP पते हैं?

e. आपके द्वारा उपयोग किए जाने वाले IP पते की श्रेणी निर्धारित करने के लिए ARIN whois डेटाबेस का उपयोग करें विश्वविद्यालय।

च. वर्णन करें कि हमलावर whois डेटाबेस और nslookup टूल का उपयोग कैसे कर सकता है किसी संस्थान पर हमला करने से पहले उसकी टोह लेना।

छ. चर्चा करें कि whois डेटाबेस सार्वजनिक रूप से उपलब्ध क्यों होना चाहिए।

P19. **इस समस्या में, हम** DNS सर्वर के पदानुक्रम का पता लगाने के लिए Unix और Linux होस्ट पर उपलब्ध उपयोगी dig **टूल का उपयोग करते हैं। याद रखें कि** चित्र 2.19 में DNS क्वेरी को DNS सर्वर पर वापस भेजकर पदानुक्रम में निचले स्तर पर , DNS पदानुक्रम में एक DNS सर्वर DNS सर्वर को सौंप दिया जाता है।

client उस निचले-स्तर के DNS सर्वर का नाम बताएँ। सबसे पहले dig के लिए मैन पेज पढ़ें , और फिर निम्नलिखित प्रश्नों के उत्तर दें।

a. रूट DNS सर्वर से शुरू करना (रूट सर्वरों में से एक [am].root-servers.net से), अपने विभाग के वेब सर्वर के लिए आईपी पते हेतु प्रश्नों का एक क्रम आरंभ करें dig का उपयोग करके । आपके प्रश्न का उत्तर देते समय डेलिगेशन श्रृंखला में DNS सर्वरों के नामों की सूची दिखाएँ।

ख. कई लोकप्रिय वेब साइटों के लिए भाग (ए) को दोहराएं, जैसे कि google.com, याहू.कॉम, या अमेज़न.कॉम.

P20. मान लीजिए कि आप अपने विभाग के स्थानीय DNS सर्वर में कैश तक पहुँच सकते हैं। क्या आप मोटे तौर पर यह निर्धारित करने का कोई तरीका सुझा सकते हैं कि आपके विभाग के बाहर कौन से वेब सर्वर आपके विभाग के उपयोगकर्ताओं के बीच सबसे लोकप्रिय हैं? समझाएँ।

P21. मान लीजिए कि आपके विभाग में सभी कंप्यूटरों के लिए एक स्थानीय DNS सर्वर है।

आप एक साधारण उपयोगकर्ता हैं (यानी, नेटवर्क/सिस्टम एडमिनिस्ट्रेटर नहीं)। क्या आप यह निर्धारित कर सकते हैं कि कुछ सेकंड पहले आपके विभाग के किसी कंप्यूटर से किसी बाहरी वेबसाइट को एक्सेस किया गया था या नहीं? समझाएँ।

P22. N साथियों को Gbits की फ़ाइल वितरित करने पर विचार करें। F=15 सर्वर की अपलोड दर us=30 di=2 Mbps है, और प्रत्येक साथी की डाउनलोड दर Mbps और अपलोड दर u है। N=10, 100, और 1,000 और u=300 Kbps, 700 Kbps, और 2 Mbps के लिए, एक चार्ट तैयार करें जिसमें यह दिया गया हो कि सर्वर की अपलोड दर कितनी है।

क्लाइंट-सर्वर दोनों के लिए N और u के प्रत्येक संयोजन के लिए न्यूनतम वितरण समय वितरण और पी2पी वितरण।

P23. क्लाइंट-सर्वर आर्किटेक्चर का उपयोग करके N पीयर्स को F बिट्स की फ़ाइल वितरित करने पर विचार करें । एक तरल मॉडल मान लें जहाँ सर्वर एक साथ कई पीयर्स को संचारित कर सकता है, प्रत्येक को संचारित कर सकता है

अलग-अलग दरों पर सहकर्मी, जब तक कि संयुक्त दर यू से अधिक न हो ।

us/N≤dmin. a. मान लीजिए कि एक वितरण योजना निर्दिष्ट करें जिसका वितरण समय है एनएफ/यू.

us/N≥dmin. b. मान लीजिए कि एक वितरण योजना निर्दिष्ट करें जिसका वितरण समय है एफ/डी .

निष्कर्ष निकालें कि न्यूनतम वितरण समय सामान्यतः निम्न द्वारा दिया जाता है अधिकतम {NF/us, F/dmin}। c.

P24. P2P आर्किटेक्चर का उपयोग करके N पीयर को F बिट्स की फ़ाइल वितरित करने पर विचार करें । एक द्रव मॉडल मान लें। सरलता के लिए मान लें कि dmin बहुत बड़ा है, ताकि पीयर डाउनलोड बैंडविड्थ कभी भी एक न हो अड़चन.

us≤(us+u1+...+uN)/N. a. मान लीजिए कि एक वितरण योजना निर्दिष्ट करें जिसका वितरण है एफ/यू का समय .

us≥(us+u1+...+uN)/N. b. मान लीजिए कि एक वितरण योजना निर्दिष्ट करें जिसका वितरण समय NF/(us+u1+...+uN) है।

सी. निष्कर्ष निकालें कि न्यूनतम वितरण समय सामान्यतः max{F/us, NF/(us+u1+...+uN)} द्वारा दिया जाता है।

P25. N सक्रिय साथियों के साथ एक ओवरले नेटवर्क पर विचार करें , जिसमें साथियों की प्रत्येक जोड़ी में एक सक्रिय TCP कनेक्शन है। इसके अतिरिक्त, मान लीजिए कि TCP कनेक्शन कुल M राउटर से होकर गुजरते हैं। संबंधित ओवरले नेटवर्क में कितने नोड और किनारे हैं?

P26. मान लीजिए बॉब बिटटोरेंट टोरेंट से जुड़ता है, लेकिन वह किसी अन्य पीयर पर कोई डेटा अपलोड नहीं करना चाहता (तथाकथित फ्री-राइडिंग)।

ए. बॉब का दावा है कि वह झुंड द्वारा साझा की गई फ़ाइल की पूरी कॉपी प्राप्त कर सकता है। क्या बॉब का दावा संभव है? क्यों या क्यों नहीं?

बी. बॉब आगे दावा करता है कि वह अपने विभाग की कंप्यूटर लैब में कई कंप्यूटरों (अलग-अलग आईपी पते के साथ) के संग्रह का उपयोग करके अपनी "मुफ़्त-सवारी" को और अधिक कुशल बना सकता है। वह ऐसा कैसे कर सकता है?

P27. एक DASH सिस्टम पर विचार करें जिसके लिए N वीडियो संस्करण (N अलग-अलग दरों और गुणवत्ता पर) और N ऑडियो संस्करण (N अलग-अलग दरों और गुणवत्ता पर) हैं। मान लीजिए कि हम अनुमति देना चाहते हैं

प्लेयर को किसी भी समय N वीडियो संस्करणों में से किसी एक और N ऑडियो संस्करणों में से किसी एक को चुनने की अनुमति होगी।

a. यदि हम ऐसी फाइल बनाते हैं जिसमें ऑडियो वीडियो के साथ मिल जाता है, तो सर्वर केवल एक ही फाइल भेजता है। किसी निश्चित समय पर मीडिया स्ट्रीम के लिए, सर्वर को कितनी फ़ाइलों को संग्रहीत करने की आवश्यकता होगी (प्रत्येक का अलग URL)?

ख. यदि सर्वर ऑडियो और वीडियो स्ट्रीम को अलग-अलग भेजता है और क्लाइंट के पास स्ट्रीम्स को सिंक्रनाइज़ करने के लिए, सर्वर को कितनी फ़ाइलों को संग्रहीत करने की आवश्यकता होगी?

P28. एक होस्ट पर पायथन प्रोग्राम TCPClient और UDPClient तथा दूसरे होस्ट पर TCPServer और UDPServer को स्थापित और संकलित करें।

क. मान लीजिए कि आप TCPServer चलाने से पहले TCPClient चलाते हैं। क्या होगा? क्यों? ख. मान लीजिए कि आप UDPServer चलाने से पहले UDPClient चलाते हैं। क्या होगा? क्यों? ग. यदि आप क्लाइंट और सर्वर साइड के लिए अलग-अलग पोर्ट नंबर का उपयोग करते हैं तो क्या होगा?

P29. मान लीजिए कि UDPClient.py में, सॉकेट बनाने के बाद, हम यह पंक्ति जोड़ते हैं:

```
clientSocket.बाइंड(('', 5432))
```

क्या UDPServer.py को बदलना ज़रूरी हो जाएगा? UDPClient और UDPServer में सॉकेट के लिए पोर्ट नंबर क्या हैं? यह बदलाव करने से पहले वे क्या थे?

P30. क्या आप अपने ब्राउज़र को एक वेब साइट पर एक साथ कई कनेक्शन खोलने के लिए कॉन्फ़िगर कर सकते हैं? एक साथ बड़ी संख्या में TCP होने के क्या लाभ और नुकसान हैं? कनेक्शन?

P31. हमने देखा है कि इंटरनेट TCP सॉकेट भेजे जा रहे डेटा को बाइट स्ट्रीम के रूप में मानते हैं लेकिन UDP सॉकेट संदेश सीमाओं को पहचानते हैं। बाइट-ओरिएंटेड API का एक लाभ और एक नुकसान क्या है, जबकि API स्पष्ट रूप से एप्लिकेशन-परिभाषित संदेश सीमाओं को पहचानता और संरक्षित करता है?

P32. अपाचे वेब सर्वर क्या है? इसकी कीमत कितनी है? वर्तमान में इसमें क्या कार्यक्षमता है? इस प्रश्न का उत्तर पाने के लिए आप विकिपीडिया पर नज़र डाल सकते हैं।

सॉकेट प्रोग्रामिंग असाइनमेंट

कम्पेनियन वेबसाइट में छह सॉकेट प्रोग्रामिंग असाइनमेंट शामिल हैं। पहले चार असाइनमेंट नीचे संक्षेप में दिए गए हैं। पाँचवाँ असाइनमेंट ICMP प्रोटोकॉल का उपयोग करता है और अंत में संक्षेप में दिया गया है

अध्याय 5 का **छठा असाइनमेंट मल्टीमीडिया प्रोटोकॉल का उपयोग करता है और** अध्याय 9 के अंत में इसका सारांश दिया गया है। यह अत्यधिक अनुशंसित है कि छात्र इनमें से कई, यदि सभी नहीं, तो असाइनमेंट पूरा करें।

छात्र इन असाइनमेंट का पूरा विवरण, साथ ही पायथन कोड के महत्वपूर्ण स्निपेट यहां पा सकते हैं वेब साइट www.pearsonhighered.com/cs-resources.

असाइनमेंट 1: वेब सर्वर

इस असाइनमेंट में, आप पायथन में एक सरल वेब सर्वर विकसित करेंगे जो केवल एक अनुरोध को संसाधित करने में सक्षम है। विशेष रूप से, आपका वेब सर्वर (i) क्लाइंट (ब्राउज़र) द्वारा संपर्क किए जाने पर एक कनेक्शन सॉकेट बनाएगा; (ii) इस कनेक्शन से HTTP अनुरोध प्राप्त करेगा; (iii) अनुरोध की जा रही विशिष्ट फ़ाइल को निर्धारित करने के लिए अनुरोध को पार्स करेगा; (iv) सर्वर के फ़ाइल सिस्टम से अनुरोधित फ़ाइल प्राप्त करेगा; (v) हेडर लाइनों से पहले अनुरोधित फ़ाइल से युक्त एक HTTP प्रतिक्रिया संदेश बनाएगा; और (vi) अनुरोध करने वाले ब्राउज़र को TCP कनेक्शन पर प्रतिक्रिया भेजेगा। यदि कोई ब्राउज़र ऐसी फ़ाइल का अनुरोध करता है जो आपके सर्वर में मौजूद नहीं है, तो आपके सर्वर को "404 नहीं मिला" त्रुटि संदेश वापस करना चाहिए।

कंपेनियन वेबसाइट में, हम आपके सर्वर के लिए कंकाल कोड प्रदान करते हैं। आपका काम कोड को पूरा करना, अपना सर्वर चलाना और फिर अलग-अलग होस्ट पर चल रहे ब्राउज़रों से अनुरोध भेजकर अपने सर्वर का परीक्षण करना है। यदि आप अपने सर्वर को ऐसे होस्ट पर चलाते हैं जिस पर पहले से ही एक वेब सर्वर चल रहा है, तो आपको अपने वेब सर्वर के लिए पोर्ट 80 के अलावा किसी दूसरे पोर्ट का उपयोग करना चाहिए।

असाइनमेंट 2: यूडीपी पिंगर

इस प्रोग्रामिंग असाइनमेंट में, आप पायथन में एक क्लाइंट पिंग प्रोग्राम लिखेंगे। आपका क्लाइंट एक पिंग भेजेगा।
सर्वर को सरल पिंग संदेश, सर्वर से संबंधित पोंग संदेश प्राप्त करना, और
क्लाइंट द्वारा पिंग संदेश भेजने और पोंग संदेश प्राप्त करने के बीच की देरी का निर्धारण करें।
इस देरी को राउंड ट्रिप टाइम (RTT) कहा जाता है। क्लाइंट और सर्वर द्वारा प्रदान की जाने वाली कार्यक्षमता आधुनिक ऑपरेटिंग सिस्टम में उपलब्ध मानक पिंग प्रोग्राम द्वारा प्रदान की जाने वाली कार्यक्षमता के समान है।
हालाँकि, मानक पिंग प्रोग्राम इंटरनेट कंट्रोल मैसेज प्रोटोकॉल (ICMP) का उपयोग करते हैं (जिसे हम
अध्याय 5 में अध्ययन)। यहां हम एक गैर-मानक (लेकिन सरल!) UDP-आधारित पिंग प्रोग्राम बनाएंगे।

आपका पिंग प्रोग्राम UDP पर लक्ष्य सर्वर को 10 पिंग संदेश भेजना है। प्रत्येक संदेश के लिए, आपका
क्लाइंट को RTT निर्धारित करना और प्रिंट करना है जब संबंधित पोंग संदेश वापस आ जाता है। क्योंकि
यूडीपी एक अविश्वसनीय प्रोटोकॉल है, क्लाइंट या सर्वर द्वारा भेजा गया पैकेट खो सकता है। इस कारण से,
क्लाइंट पिंग संदेश के उत्तर के लिए अनिश्चित काल तक प्रतीक्षा नहीं कर सकता। आपको क्लाइंट को सर्वर से उत्तर के लिए एक सेकंड तक प्रतीक्षा करवाना चाहिए; यदि कोई उत्तर प्राप्त नहीं होता है, तो क्लाइंट को यह मान लेना चाहिए कि पैकेट प्राप्त हो गया है।
खो जाने पर उसके अनुसार संदेश मुद्रित करें।

इस असाइनमेंट में, आपको सर्वर के लिए पूरा कोड दिया जाएगा (जो कंपेनियन वेबसाइट पर उपलब्ध है)। आपका काम क्लाइंट कोड लिखना है, जो सर्वर कोड से काफी मिलता-जुलता होगा। यह अनुशंसा की जाती है कि आप पहले सर्वर कोड का ध्यानपूर्वक अध्ययन करें। फिर आप अपना क्लाइंट कोड लिख सकते हैं, सर्वर कोड से उदारतापूर्वक लाइनें काटकर चिपका सकते हैं।

असाइनमेंट 3: मेल क्लाइंट

इस प्रोग्रामिंग असाइनमेंट का लक्ष्य एक सरल मेल क्लाइंट बनाना है जो किसी भी प्राप्तकर्ता को ई-मेल भेजता है। आपके क्लाइंट को एक मेल सर्वर (जैसे, एक Google मेल सर्वर) के साथ एक TCP कनेक्शन स्थापित करना होगा, SMTP प्रोटोकॉल का उपयोग करके मेल सर्वर के साथ संवाद करना होगा, प्राप्तकर्ता को एक ई-मेल संदेश भेजना होगा

(उदाहरण के लिए, अपने मित्र) को मेल सर्वर के माध्यम से संदेश भेजें, और अंत में मेल सर्वर के साथ TCP कनेक्शन बंद कर दें।

इस असाइनमेंट के लिए, कंपेनियन वेबसाइट आपके क्लाइंट के लिए स्केलेटन कोड प्रदान करती है। आपका काम कोड को पूरा करना और अलग-अलग उपयोगकर्ता खातों को ई-मेल भेजकर अपने क्लाइंट का परीक्षण करना है। आप अलग-अलग सर्वर (उदाहरण के लिए, Google मेल सर्वर और अपने विश्वविद्यालय मेल सर्वर के माध्यम से) के माध्यम से भेजने का भी प्रयास कर सकते हैं।

असाइनमेंट 4: मल्टी-थ्रेडेड वेब प्रॉक्सी

इस असाइनमेंट में, आप एक वेब प्रॉक्सी विकसित करेंगे। जब आपका प्रॉक्सी ब्राउज़र से किसी ऑब्जेक्ट के लिए HTTP अनुरोध प्राप्त करता है, तो यह उसी ऑब्जेक्ट के लिए एक नया HTTP अनुरोध उत्पन्न करता है और इसे मूल सर्वर पर भेजता है। जब प्रॉक्सी को मूल सर्वर से ऑब्जेक्ट के साथ संबंधित HTTP प्रतिक्रिया प्राप्त होती है, तो यह ऑब्जेक्ट सहित एक नया HTTP प्रतिक्रिया बनाता है और इसे क्लाइंट को भेजता है। यह प्रॉक्सी मल्टी-थ्रेडेड होगी, ताकि यह एक ही समय में कई अनुरोधों को संभालने में सक्षम हो।

इस असाइनमेंट के लिए, कम्पेनियन वेबसाइट प्रॉक्सी सर्वर के लिए कंकाल कोड प्रदान करती है। आपका काम कोड को पूरा करना है, और फिर अलग-अलग ब्राउज़रों से आपके माध्यम से वेब ऑब्जेक्ट का अनुरोध करके इसका परीक्षण करना है।
प्रॉक्सी.

वायरशार्क लैब: HTTP

लैब 1 में Wireshark पैकेट स्निफ़र के साथ अपने पैर भिगोने के बाद, अब हम ऑपरेशन में प्रोटोकॉल की जांच करने के लिए Wireshark का उपयोग करने के लिए तैयार हैं। इस लैब में, हम HTTP प्रोटोकॉल के कई पहलुओं का पता लगाएंगे: बुनियादी GET/रिप्लाई इंटरैक्शन, HTTP संदेश प्रारूप, बड़ी HTML फ़ाइलों को पुनर्प्राप्त करना, एम्बेडेड URL के साथ HTML फ़ाइलों को पुनर्प्राप्त करना, लगातार और गैर-स्थायी कनेक्शन, और HTTP प्रमाणीकरण और सुरक्षा।

जैसा कि सभी वायरशार्क प्रयोगशालाओं के मामले में होता है, इस प्रयोगशाला का पूरा विवरण इस पुस्तक की वेब साइट पर उपलब्ध है,

www.pearsonhighered.com/cs-resources.

वायरशार्क लैब: DNS

इस प्रयोगशाला में, हम DNS के क्लाइंट पक्ष पर करीब से नज़र डालेंगे, जो कि इंटरनेट को क्लाइंट के साथ संवाद करने वाला प्रोटोकॉल है।

होस्टनाम से लेकर आईपी एड्रेस तक। सेक्शन 2.5 **से याद करें** कि DNS में क्लाइंट की भूमिका अपेक्षाकृत सरल है - क्लाइंट अपने स्थानीय DNS सर्वर को क्वेरी भेजता है और वापस प्रतिक्रिया प्राप्त करता है। DNS क्लाइंट के लिए अदृश्य रूप से बहुत कुछ हो सकता है, क्योंकि पदानुक्रमित DNS सर्वर क्लाइंट की DNS क्वेरी को पुनरावर्ती या पुनरावृत्त रूप से हल करने के लिए एक दूसरे के साथ संवाद करते हैं। हालाँकि, DNS क्लाइंट के दृष्टिकोण से, प्रोटोकॉल काफी सरल है - स्थानीय DNS सर्वर के लिए एक क्वेरी तैयार की जाती है और एक प्रतिक्रिया होती है

उस सर्वर से प्राप्त जानकारी। हम इस प्रयोगशाला में DNS को क्रियाशील अवस्था में देखते हैं।

जैसा कि सभी वायरशार्क प्रयोगशालाओं के मामले में होता है, इस प्रयोगशाला का पूरा विवरण इस पुस्तक की वेब साइट पर उपलब्ध है,

www.pearsonhighered.com/cs-resources.

एक साक्षात्कार...

मार्क आंद्रेसेन

मार्क आंद्रेसेन मोजेक के सह-निर्माता हैं, यह वेब ब्राउज़र है जिसने 1993 में वर्ल्ड वाइड वेब को लोकप्रिय बनाया। मोजेक में एक साफ-सुथरा, आसानी से समझ में आने वाला इंटरफ़ेस था और यह पहला ब्राउज़र था जो टेक्स्ट के साथ इन-लाइन इमेज दिखाता था। 1994 में, मार्क आंद्रेसेन और जिम क्लार्क ने नेटस्केप की स्थापना की, जिसका ब्राउज़र 1990 के दशक के मध्य तक सबसे लोकप्रिय ब्राउज़र था। नेटस्केप ने सिक्योर सॉकेट लेयर (SSL) प्रोटोकॉल और कई इंटरनेट सर्वर उत्पाद भी विकसित किए, जिनमें मेल सर्वर और SSL-आधारित वेब सर्वर शामिल हैं। अब वे वेंचर कैपिटल फर्म आंद्रेसेन होरोविट्ज़ के सह-संस्थापक और सामान्य भागीदार हैं, जो Facebook, Foursquare, Groupon, Jawbone, Twitter और Zynga जैसी होल्डिंग्स के साथ पोर्टफोलियो विकास की देखरेख करते हैं। वे बम्प, eBay, ग्लैम मीडिया, Facebook और Hewlett-Packard सहित कई बोर्डों में काम करते हैं। उनके पास Urbana-Champaign में इलिनोइस विश्वविद्यालय से कंप्यूटर विज्ञान में BS की डिग्री है।



कंप्यूटिंग में आपकी रुचि कैसे पैदा हुई? क्या आपको हमेशा से पता था कि आप इसी क्षेत्र में काम करना चाहते हैं?

सूचान प्रौद्योगिकी?

वीडियो गेम और पर्सनल कंप्यूटिंग क्रांति ठीक उसी समय आई जब मैं बड़ा हो रहा था - पर्सनल 70 के दशक के अंत और 80 के दशक की शुरुआत में कंप्यूटिंग नई तकनीक की सीमा थी। और यह सिर्फ़ Apple और IBM PC ही नहीं था, बल्कि Commodore और Atari जैसी सैकड़ों नई कंपनियाँ भी थीं। मैंने 10 साल की उम्र में “इंस्टेंट फ़्रीज़-ड्राइड बेसिक” नामक एक किताब से खुद को प्रोग्रामिंग करना सीखा और 12 साल की उम्र में अपना पहला कंप्यूटर (TRS-80 कलर कंप्यूटर - इसे देखें!) खरीदा।

कृपया अपने करियर के दौरान आपके द्वारा किए गए सबसे रोमांचक प्रोजेक्ट्स में से एक या दो का वर्णन करें।

सबसे बड़ी चुनौतियां क्या थीं?

निस्संदेह सबसे रोमांचक परियोजना '92-'93 में मूल मोज़ेक वेब ब्राउज़र थी - और उस समय सबसे बड़ी चुनौती थी किसी को भी इसे गंभीरता से लेना। उस समय, हर कोई सोचा था कि इंटरैक्टिव भविष्य बड़ी कंपनियों द्वारा "इंटरैक्टिव टेलीविजन" के रूप में पेश किया जाएगा, न कि स्टार्टअप द्वारा इंटरनेट के रूप में।

नेटवर्किंग और इंटरनेट के भविष्य के बारे में आपको क्या उत्साहित करता है? आपकी सबसे बड़ी चिंताएँ क्या हैं?

चिंताएँ?

सबसे रोमांचक बात यह है कि अनुप्रयोगों और सेवाओं की विशाल अज्ञात सीमा है। प्रोग्रामर और उद्यमी खोज करने में सक्षम हैं - इंटरनेट ने रचनात्मकता को उन्मुक्त कर दिया है मुझे नहीं लगता कि हमने पहले कभी ऐसा स्तर देखा है। मेरी सबसे बड़ी चिंता अनपेक्षित परिणामों के सिद्धांत से जुड़ी है - हम हमेशा यह नहीं जानते कि हम जो करते हैं उसका क्या परिणाम होगा, जैसे कि सरकारों द्वारा नागरिकों पर निगरानी के नए स्तर को चलाने के लिए इंटरनेट का उपयोग करना।

वेब प्रौद्योगिकी की प्रगति के संबंध में क्या कोई विशेष बात है जिसके बारे में छात्रों को जागरूक होना चाहिए?

परिवर्तन की दर - सीखने के लिए सबसे महत्वपूर्ण बात यह है कि कैसे सीखें - विशिष्ट प्रौद्योगिकियों में परिवर्तनों के प्रति लचीले ढंग से अनुकूलन कैसे करें, तथा अपने कैरियर में आगे बढ़ते हुए नए अवसरों और संभावनाओं के प्रति खुले दिमाग कैसे रखें।

पेशेवर तौर पर किन लोगों ने आपको प्रेरित किया?

वेन्नेवर बुश, टेड नेल्सन, डग एंजेलबार्ट, नोलन बुशनेल, बिल हेवलेट और डेव पैकार्ड, केन ऑलसेन, स्टीव जॉब्स, स्टीव वोज़नियाक, एंडी ग्रोव, ग्रेस हॉपर, हेडी लैमर, एलन ट्यूरिंग, रिचर्ड स्टालमैन.

कंप्यूटिंग और कंप्यूटर विज्ञान में करियर बनाने के इच्छुक छात्रों के लिए आपकी क्या सिफारिशें हैं?

सूचान प्रौद्योगिकी?

जितना संभव हो सके उतना गहराई से समझें कि प्रौद्योगिकी कैसे बनाई जाती है, और फिर यह भी सीखें कि व्यवसाय कैसे काम करता है।

क्या प्रौद्योगिकी विश्व की समस्याओं का समाधान कर सकती है?

नहीं, लेकिन हम आर्थिक विकास के माध्यम से लोगों के जीवन स्तर को उन्नत करते हैं, और इतिहास में अधिकांश आर्थिक विकास प्रौद्योगिकी से आया है - इसलिए यह सबसे अच्छा है।

अध्याय 3 परिवहन परत

एप्लीकेशन और नेटवर्क परतों के बीच स्थित ट्रांसपोर्ट लेयर, लेयर्ड नेटवर्क आर्किटेक्चर का एक केंद्रीय हिस्सा है। अलग-अलग होस्ट पर चल रही एप्लीकेशन प्रक्रियाओं को सीधे संचार सेवाएँ प्रदान करने में इसकी महत्वपूर्ण भूमिका है। इस अध्याय में हम जो शैक्षणिक दृष्टिकोण अपनाते हैं, वह ट्रांसपोर्ट-लेयर सिद्धांतों की चर्चा और मौजूदा प्रोटोकॉल में इन सिद्धांतों को कैसे लागू किया जाता है, इस पर चर्चा के बीच बारी-बारी से काम करना है; हमेशा की तरह, इंटरनेट प्रोटोकॉल, विशेष रूप से TCP और UDP ट्रांसपोर्ट-लेयर प्रोटोकॉल पर विशेष जोर दिया जाएगा।

हम ट्रांसपोर्ट और नेटवर्क लेयर के बीच संबंधों पर चर्चा करके शुरू करेंगे। यह ट्रांसपोर्ट लेयर के पहले महत्वपूर्ण कार्य की जांच करने के लिए मंच तैयार करता है - दो एंड सिस्टम के बीच नेटवर्क लेयर की डिलीवरी सेवा को एंड सिस्टम पर चलने वाली दो एप्लिकेशन-लेयर प्रक्रियाओं के बीच डिलीवरी सेवा तक विस्तारित करना। हम इंटरनेट के कनेक्शन रहित ट्रांसपोर्ट प्रोटोकॉल, UDP के अपने कवरेज में इस फ़ंक्शन को स्पष्ट करेंगे।

इसके बाद हम सिद्धांतों पर लौटेंगे और कंप्यूटर नेटवर्किंग की सबसे बुनियादी समस्याओं में से एक का सामना करेंगे - कैसे दो संस्थाएं एक ऐसे माध्यम पर विश्वसनीय ढंग से संवाद कर सकती हैं, जो डेटा को नष्ट या दूषित कर सकता है।
लगातार जटिल (और यथार्थवादी!) परिदृश्यों की एक श्रृंखला के माध्यम से, हम तकनीकों की एक श्रृंखला का निर्माण करेंगे जिसका उपयोग परिवहन प्रोटोकॉल इस समस्या को हल करने के लिए करते हैं। फिर हम दिखाएंगे कि ये सिद्धांत TCP, इंटरनेट के कनेक्शन-उन्मुख परिवहन प्रोटोकॉल में कैसे सन्निहित हैं।

अब हम नेटवर्किंग में दूसरी मूलभूत रूप से महत्वपूर्ण समस्या पर आगे बढ़ेंगे - नेटवर्क के भीतर भीड़भाड़ से बचने या उससे उबरने के लिए ट्रांसपोर्ट-लेयर इकाइयों की संचरण दर को नियंत्रित करना। हम भीड़भाड़ के कारणों और परिणामों के साथ-साथ आम तौर पर इस्तेमाल की जाने वाली भीड़भाड़-नियंत्रण तकनीकों पर विचार करेंगे। भीड़भाड़ नियंत्रण के पीछे के मुद्दों की ठोस समझ प्राप्त करने के बाद, हम भीड़भाड़ नियंत्रण के लिए TCP के दृष्टिकोण का अध्ययन करेंगे।